RANCANG BANGUN APLIKASI BENGKEL TERDEKAT MENGGUNAKAN *GOOGLE MAPS API* PADA *SMART PHONE* ANDROID

SKRIPSI

REKAYASA PERANGKAT LUNAK

Diajukan untuk Memenuhi Persyaratan
Memperoleh Gelar Sarjana Komputer

Disusun Oleh :
RICO MAULANA A
0910683081

PROGRAM STUDI TEKNIK INFORMATIKA
PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
2013

LEMBAR PERSETUJUAN

RANCANG BANGUN APLIKASI BENGKEL TERDEKAT
MENGGUNAKAN *GOOGLE MAPS API*
PADA *SMART PHONE* ANDROID

Disusun oleh :

RICO MAULANA A

NIM. 0910683081

Telah diperiksa dan disetujui oleh

Dosen Pembimbing I	Dosen Pembimbing II
Aryo Pinandito, ST., M.MT	**Barlian Henryranu P, S.T., M.T.**
NIK. 83051916110374	**NIK. 821024 06 1 1 0254**

LEMBAR PENGESAHAN

RANCANG BANGUN APLIKASI BENGKEL TERDEKAT MENGGUNAKAN *GOOGLE MAPS API* PADA *SMART PHONE* ANDROID

SKRIPSI

KONSENTRASI REKAYASA PERANGKAT LUNAK

Diajukan untuk memenuhi persyaratan memperoleh gelar Sarjana Komputer

Disusun oleh :

RICO MAULANA A

NIM. 0910683081

Skripsi ini telah diuji dan dinyatakan lulus pada

tanggal 18 Juli 2014

Penguji I	Penguji II
Denny Sagita Rusdianto, S.Kom., M.Kom.	**Fajar Pradana, S.ST,.M.Eng**
NIK. 85112406110250	**NIK. 87112116110371**

Penguji III

Aswin Suharsono, ST., MT.

NIK. 84091906110251

Mengetahui

Ketua Program Studi Informatika

Drs. Marji, MT.

NIP. 19670801 199203 1 001

PERNYATAAN ORISINALITAS SKRIPSI

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah SKRIPSI ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis dikutip dalam naskah ini dan disebutkan dalam sumber kutipan dan daftar pustaka.

Apabila ternyata didalam naskah SKRIPSI ini dapat dibuktikan terdapat unsur-unsur PLAGIASI, saya bersedia SKRIPSI ini digugurkan dan gelar akademik yang telah saya peroleh (SARJANA) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku. (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 27 Mei 2014

Mahasiswa,

Rico Maulana A

NIM 0910683081

Kata Pengantar

Puji syukur penulis panjatkan kehadirat Tuhan Yang Maha Esa karena hanya dengan rahmat dan karunia-Nya, penulis dapat menyelesaikan proposal skripsi dengan judul "Rancang bangun Aplikasi Bengkel berbasis *Google Maps API* pada *SmartPhone* Android.

Melalui kesempatan ini, penulis ingin menyampaikan rasa hormat dan terima kasih yang sebesar-besarnya kepada semua pihak yang telah memberikan bantuan dan dukungan selama penulisan proposal skripsi, diantaranya:

1. Kedua Orang Tua (H. Moh. Syahid dan Hj. ST. Sulaiha), dan seluruh keluarga yang senantiasa tiada henti hentinya memberikan do'a demi terselesainya skripsi ini.
2. Bapak Ir. Sutrisno, M.T, Bapak Ir. Heru Nurwasito, M.Kom, Bapak Himawat Aryadita, S.T, M.Sc, dan Bapak Eddy Santoso, S.Kom selaku Ketua, Wakil Ketua 1, Wakil Ketua 2 dan Wakil Ketua 3 Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya.
3. Bapak Drs. Marji, MT dan Bapak Issa Arwani, S.Kom., M.Sc selaku Ketua dan Sekretaris Program Studi Informatika serta segenap Bapak/Ibu Dosen, Staff Administrasi dan Perpustakaan Program Studi Informatika Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya
4. Bapak Aryo Pinandito, ST., M.MT. selaku dosen pembimbing I yang telah memberikan ilmu dan saran untuk proposal skripsi ini.
5. Bapak Barlian Henryranu Prasetio, ST., M.T. selaku dosen pembimbing II yang juga memberikan ilmu dan saran untuk proposal skripsi ini.
6. Chandra S.Kom, Arief Azwar Alvian S.Kom, Hanas Subakti S.Kom, Arga Suwastika S.Kom, A.Yazid B S.Kom, Adhit Khoteb S.Kom, Betha Nila. D S.E, Rakanta Rifky S.Kom, Nizam, Erwin S.Kom, Bagas S.Kom, dan
7. Teman-teman 2009 Teknik Informatika, terimakasih atas segala bantuannya selama menempuh studi di Teknik Informatika Universitas Brawijaya.

8. Semua pihak yang tidak dapat penulis sebutkan satu per satu yang terlibat baik secara langsung maupun yang tidak langsung demi terselesaikannya skripsi ini.

Penulis sadar bahwa proposal skripsi ini masih banyak kekurangan, oleh karena itu kritik dan saran yang bersifat membangun sangat diharapkan untuk menyempurnakan proposal skripsi ini.Penulis berharap proposal skripsi ini dapat bermanfaat khususnya bagi diri sendiri dan bagi semua pihak.

Malang, Mei 2014

Penulis

ABSTRAK

Rico Maulana A. 2014. Rancang bangun Aplikasi Bengkel berbasis *Google Maps API* pada *SmartPhone* Android. Skripsi Program Study Teknik Informatika, Program Teknologi Informasi dan Ilmu Komputer, Universitas Brawijaya.

Dosen Pembimbing : Aryo Pinandito, ST., M.MT dan Barlian Henryranu Prasetio, ST., M.T.

Mengikuti perkembangan sistem informasi yang begitu pesat saat ini, dan fasilitas yang tidak mudah dijangkau menyebabkan sulitnya pemenuhan kebutuhan sehari-hari, sebagai salah contoh yaitu kurangnya efisien dalam pencarian secara manual atau pencarian tempat-tempat tertentu seperti pencarian lokasi bengkel. Karena, sebagian besar masyarakat ketika mengalami kerusakan kendaraan ataupun ban bocor hanya berjalan searah dengan tujuannya untuk mendapatkan bengkel, sebenarnya tidak menutup kemungkin untuk berbalik arah bisa mendapatkan bengkel yang lebih dekat. Kemudian, ketika ditemukan persimpangan jalan maka pengguna tidak tahu harus pergi kearah mana untuk mendapatkan lokasi bengkel yang terdekat. Oleh sebab itu penulis merasa perlu adanya sistem yang bisa menjangkau pencarian secara otomatis dengan sederhana. Aplikasi bengkel merupakan aplikasi berbasis *mobile phone* sederhana yang mudah diakses oleh pengguna sebagai sarana pencarian otomatis untuk mencari lokasi bengkel terdekat dengan lokasi pengguna saat itu, sehingga memudahkan pengguna ketika membutuhkan. Jadi aplikasi bengkel merupakan aplikasi yang dapat memenuhi pencarian otomatis untuk mencari lokasi bengkel terdekat dengan lokasi pengguna saat itu.

Kata Kunci : Android, Mobil phone

ABSTRACT

Rico Maulana A. 2014. *Repair Shop Design Application Based on Google Maps on Android Smartphone*

***Adviser* : Aryo Pinandito, ST., M.MT dan Barlian Henryranu Prasetio, ST., M.T.**

Following the rapidly of information development today, and the difficulty of daily needs compliance due to the facilities are quite difficult to access, as an example the lack of efficient in search of certain place , such as the repair shop location. Because mostly peoples when had a tire leak are just walks straight to the destination to go to the repair shop without considering to turn way back to get to the closer repair shop.Moreover when get into crossroad, users doesn't know where to go to the closer repair shop location. Base on that study, author felt the need for a system that can reach out automatically with a simple search. Repair shop application is a mobile phoned-based applications that are easily accessible by users which allow users to find nearest garage based on users current location. The conclusion is repair shop application can reach automatic searching to define the nearest repair shop location based on user current location.

Keyword : Android, Mobil phone

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Mengikuti perkembangan sistem informasi yang begitu pesat saat ini, dan fasilitas yang tidak mudah dijangkau menyebabkan sulitnya pemenuhan kebutuhan sehari-hari, sebagai salah satu contoh yaitu kurangnya efisien dalam pencarian secara otomatis atau pencarian tempat-tempat tertentu seperti pencarian lokasi “bengkel”. Bengkel merupakan tempat yang tidak sering kita kunjungi, tetapi bisa sangat bermanfaat ketika kita mengalami kerusakan kendaraan ataupun ban bocor saat perjalanan. Kurangnya efisien dalam pencarian bengkel secara otomatis ini menyebabkan pengguna merasa kesulitan untuk mendapatkan lokasi bengkel terdekat dengan lokasinya ketika mengalami kendala kerusakan kendaraan dan berjumpa dengan persimpangan jalan yang tidak tahu harus pergi kearah mana untuk mendapatkan lokasi bengkel yang terdekat. Karena, dari sekian banyak pengguna yang mengalami kerusakan kendaraan atau ban bocor, pengguna hanya melakukan pencarian secara manual saja yaitu berjalan searah dengan tujuannya untuk mendapatkan bengkel. Sebenarnya, tidak menutup kemungkinan ketika berbalik arah pengguna bisa menemukan bengkel yang lebih dekat dibanding harus berjalan searah tujuannya yang lebih jauh untuk mendapatkan bengkel. Maka oleh sebab itu penulis merasa perlu adanya sistem yang bisa menjangkau pencarian secara otomatis dengan sederhana.

Aplikasi bengkel terdekat merupakan aplikasi berbasis *mobile phone* sederhana yang mudah diakses oleh pengguna sebagai sarana pencarian otomatis untuk mencari lokasi bengkel terdekat dengan lokasi pengguna saat itu. Kemudahan spesifik yang diberikan oleh aplikasi ini yaitu, memberi kemudahan bagi pengguna untuk mencari lokasi bengkel terdekat dengan tempat kejadian pengguna saat itu. Sehingga pengguna tidak harus berjalan jauh yang searah dengan tujuannya untuk menemukan lokasi bengkel yang diperlukannya. Pada aplikasi ini juga disediakan informasi / kontak ditiap-tiap lokasi bengkel yang bisa dihubungi oleh pengguna.

1.2 Rumusan Masalah

Berdasarkan uraian latar belakang di atas, maka dapat diambil rumusan masalah pada penelitian ini yaitu bagaimana menganalisis, merancang dan mengimplementasikan sebuah aplikasi yang dapat menampilkan informasi bengkel terdekat dengan lokasi pengguna yang diterapkan pada *mobile phone* berbasis Android sehingga dapat memberikan kemudahan bagi pengguna ketika mengalami kerusakan kendaraan.

1.3 Batasan Masalah

Agar permasalahan yang dirumuskan dapat lebih terfokus, maka pada penelitian ini dibatasi dalam hal:

1. Data lokasi bengkel yang digunakan adalah data bengkel di kota Malang dan kota Pamekasan Madura.
2. Tidak membahas sistem satelit GPS dan jaringan selular.
3. Tidak membahas keamanan pada saat mengakses *Google Maps* API.

1.4 Tujuan

Tujuan yang ingin dicapai dalam penelitian ini yaitu membuat aplikasi yang dapat mencari lokasi bengkel terdekat dengan lokasi pengguna saat itu di kota Malang dan kota Pamekasan Madura pada perangkat *mobile phone* dengan penentuan posisi pengguna menggunakan GPS.

1.5 Manfaat

Penulisan skripsi ini diharapkan mempunyai manfaat yang baik dan berguna bagi pembaca dan penulis. Adapun manfaat yang diharapkan dalam penelitian ini yaitu, untuk memberikan kemudahan bagi pengguna ketika mencari lokasi bengkel terdekat dengan lokasi pengguna saat itu khususnya di kota Malang dan kota Pamekasan Madura.

1.6 Sistematika Penulisan

Sistematika penulisan dalam skripsi ini sebagai berikut:

BAB I Pendahuluan

Memuat latar belakang, rumusan masalah, tujuan, batasan masalah, manfaat, metodologi pembahasan, dan sistematika penulisan.

BAB II Kajian Pustaka dan Dasar teori

Menguraikan tentang dasar teori dan referensi yang mendasari pembuatan aplikasi bengkel terdekat yang berbasis *Google Maps* API dengan sistem operasi Android, analisis perancangan serta strategi pengujian.

BAB III Metode Penelitian

Membahas metode yang digunakan dalam penelitian yang terdiri dari studi literatur, perancangan sistem perangkat lunak, implementasi sistem perangkat lunak, pengujian dan analisis, serta pengambilan kesimpulan.

BAB IV Analisis dan Perancangan

Membahas analisis kebutuhan dan perancangan yang sesuai dengan teori yang ada.

BAB V Implementasi

Membahas tentang implementasi dari sistem aplikasi.

BAB VI Pengujian

Memuat proses dan hasil pengujian terhadap sistem yang telah direalisasikan.

BAB VII Kesimpulan

Memuat kesimpulan yang diperoleh dari pembuatan dan pengujian perangkat lunak yag dikembangkan dalam skripsi ini serta saran-saran untuk pengembangan lebih lanjut.

BAB II
TINJAUAN PUSTAKA

Bab ini berisi tinjauan pustaka yang meliputi kajian pustaka dan dasar teori yang diperlukan untuk penelitian. Kajian pustaka adalah membahas penelitian yang telah ada dan yang diusulkan. Dasar teori adalah membahas teori yang diperlukan untuk menyusun penelitian yang diusulkan.

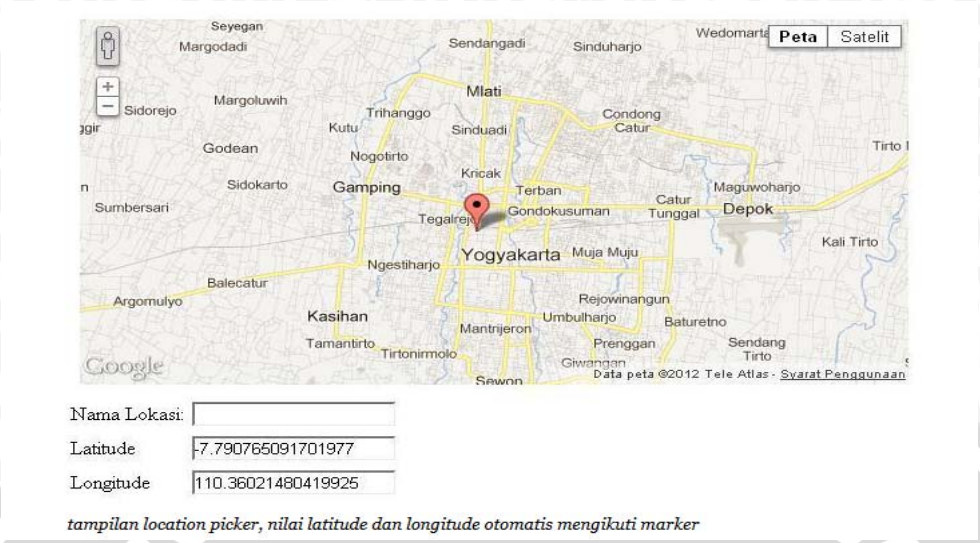
Kajian pustaka pada penelitian ini adalah membahas penelitian sebelumnya yang berjudul "Sistem Pemandu Pencarian Masjid Terdekat". Teori dasar yang akan dibahas pada bab ini yaitu konsep dasar *Global Positioning Sistem, Google Maps API* yang terdiri dari *MapView, MapActivity, OverlayItem,* dan *GeoPoint. Android Location API* yang terdiri dari *Geocoder, Location, LocationManager*, dan *LocationListener*. *Android* dan *JSON.*

2.1 Kajian Pustaka

Kajian pustaka pada penelitian ini adalah membahas penelitian sebelumnya yang berjudul "Sistem Pemandu Pencarian Masjid Terdekat". Penelitian ini membahas tentang pencarian lokasi masjid terdekat dengan menggunakan *Google Maps* API dan GPS untuk penentuan lokasi pengguna saat itu.

Perbedaan terkait pada penelitian sebelumnya terletak pada notifikasi pengingat sholat. Jadi pada aplikasi "pemandu pencarian mesjid" ini juga disediakan tampilan yang dapat menampilkan notifikasi untuk pengingat pengguna bahwa waktu sholat akan tiba. Sedangkan pada aplikasi bengkel hanya difokuskan untuk pencarian lokasi bengkel terdekat saja dengan peanmpilan rute dari lokasi pengguna ke lokasi bengkel yang diinginkan pengguna.

Google Maps API juga digunakan oleh penulis (admin) untuk alat bantu pencarian / mendapatkan *longitude* dan *latitude* penentuan lokasi bengkel / lokasi marker bengkel pada peta dengan memanfaatkan *location picker* pada *google map.* Gambar 2.1 merupakan contoh *location picker* pada *google map.*



Gambar 2.1 Location picker

Sumber [CHA-13]

Gambar 2.1 merupakan salah satu contoh untuk mendapatkan *longitude* dan *latitude* suatu tempat. Dengan hanya menggeser marker yang ada pada map maka kita akan mendapatkan nilai dari suatu tempat yang sudah kita tentukan / tandai. Pada tabel 2.1 adalah contoh kode program *location picker*.

Tabel 2.1 Contoh kode program *location picker*

```
<html>
	<head>
		<meta http-equiv="content-type"
content="text/html; charset=UTF-8" />
		<title>Google Maps Location picker</title>
		<style type="text/css">
			#map {
				margin: 10px;
				width: 600px;
				height: 300px;
				padding: 10px;
			}
</style>
<script src="http://maps.google.com/maps/api/js?sensor=false"
		type="text/javascript"></script>
	</head>
	<body>
	<div id="map"></div>
	<Tabel   >
	<form  action=''>
	<tr><td>Nama Lokasi:</td>
		<td><input type="text" name='nama_lokasi'
id='nama_lokasi'></td></tr>
```

```
        <tr><td>Latitude</td>
        <td><input type="text" name='latitude'
id='latitude'></td></tr>
<tr><td>Longitude</td>
<td><input type="text" name='longitude' id='longitude'></td></tr>
</form>
</Tabel   >


<script type="text/javascript">
                //* Fungsi untuk mendapatkan nilai latitude
longitude
function updateMarkerPosition(latLng) {
        document.getElementById('latitude').value = [latLng.lat()]
    document.getElementById('longitude').value = [latLng.lng()]
}

var map = new google.maps.Map(document.getElementById('map'), {
zoom: 12,
center: new google.maps.LatLng(-7.781921,110.364678),
 mapTypeId: google.maps.MapTypeId.ROADMAP
        });
//posisi awal marker
var latLng = new google.maps.LatLng(-7.781921,110.364678);

/* buat marker yang bisa di drag lalu
  panggil fungsi updateMarkerPosition(latLng)
 dan letakan posisi terakhir di id=latitude dan id=longitude
 */
var marker = new google.maps.Marker({
                position : latLng,
                title : 'lokasi',
                map : map,
                draggable : true
        });

updateMarkerPosition(latLng);
google.maps.event.addListener(marker, 'drag', function() {
 // ketika marker di drag, otomatis nilai latitude dan longitude
 //menyesuaikan dengan posisi marker
    updateMarkerPosition(marker.getPosition());
  });
</script>
        </body>
</html>
```

Sumber [CHA-13]

2.2 *Global Positioning Sistem (GPS)*

Global Positioning Sistem (GPS) adalah sistem untuk menentukan letak di permukaan bumi dengan bantuan penyelarasan (*synchronization*) sinyal satelit. Sistem ini menggunakan 24 satelit yang mengirimkan sinyal gelombang mikro ke Bumi. Sinyal ini diterima oleh alat penerima di permukaan, dan digunakan untuk

menentukan letak, kecepatan, arah, dan waktu. Sistem *navigasi* berbasis satelit ditempatkan ke orbit oleh Departemen Pertahanan Amerika Serikat.

GPS pada awalnya ditujukan untuk aplikasi militer, namun pada 1980-an, pemerintah membuat sistem yang tersedia untuk penggunaan sipil. GPS bekerja di semua kondisi cuaca, di mana saja di dunia, 24 jam sehari. Tidak ada biaya langganan atau biaya setup untuk menggunakan GPS. GPS juga dapat diakses dengan bebas oleh siapa saja asalkan mempunyai sebuah alat penerima GPS. Data lokasi yang dimaksudkan di sini adalah berupa posisi geografis, yaitu *latitude* (lintang), *longitude* (bujur), dan *altitude* (ketinggian). GPS dapat menghitung koordinat lintang dan bujur suatu titik dengan mengunci minimal 3 sinyal satelit yang berbeda [RRD-11:6].

Informasi GPS ditransmisikan oleh beberapa satelit (tiga satelit misalnya) sehingga GPS *receiver* mampu mengkalkulasi dan menampilkan seakurat mungkin posisi, kecepatan dan informasi waktu kepada pengguna GPS.

2.3 *Google Maps API*

Google Maps merupakan sebuah layanan peta dunia virtual berbasis web yang disediakan oleh *Google Inc*. *Google Maps* dapat dilekatkan sebagai elemen dalam tata letak antarmuka pengguna yang dirancang oleh pemrogram. Untuk menghubungkan aplikasi perangkat lunak dengan *Google Maps* diperlukan sebuah kunci yang diistilahkan sebagai *API Key*.

Google Maps adalah suatu peta dunia yang dapat kita gunakan untuk melihat suatu daerah. Dengan kata lain, *Google Maps* merupakan suatu peta yang dapat dilihat dengan menggunakan suatu *browser*. Kita dapat menambahkan fitur *Google Maps* dalam web yang telah kita buat atau pada blog kita yang berbayar maupun gratis sekalipun dengan *Google Maps API. Google Maps API* adalah suatu *library* yang berbentuk JavaScript. Pustaka eksternal yang ditawarkan oleh Google untuk menghubungkan aplikasi Android dengan *Google Maps* adalah *com.google.Android.maps*. Untuk memanfaatkan pustaka ini, diperlukan Google *API's add-on* sehingga dapat dibuat aplikasi berbasis *Google Maps* pada Android SDK dengan akses terhadap data *Google Maps*. Pustaka *Google Maps* API terdiri atas beberapa kelas. Berikut merupakan kelas-kelas tersebut [AFD-13:05].

2.3.1 *MapActivity*

MapActivity merupakan kelas abstrak yang digunakan untuk menampilkan sebuah *MapView*. Dalam pemrograman Android, untuk bisa membuat aplikasi yang mengakses layanan *Google Maps*, harus dilakukan pembuatan kelas yang merupakan kelas anak dari *MapActivity*. Di dalam kelas itu nantinya baru dilakukan pengkodean untuk menampilkan *MapView* dan melakukan pengaturan-pengaturan yang berkaitan. Pada tabel 2.2 merupakan contoh kode program *MapActivity*:

Tabel 2.2 Kode program *MapActivity*

```
public kelas TampilkanMap extends MapActivity
       implements OnCheckedChangeListener {
              MapView mp=null;
              MapController mc;
              RadioGroup rg;

              /** Called when thr activity is first created. */
              @Override
              public void onCreate (Bundle savedIstanceState) {
              super.onCreate (Bundle savedIstanceState);
              setContentView(R.layout.main);

              mp=(MapView)findViewById(R.Id.mapView);
              mp.setBuiltInZoomControls(true);
              rg=(RadioGroup)findViewById(R.Id.viewRG);
              rg.setOnCheckedChangeListener(this);
              }

              @Override
              protected boolean isRouteDisplayed(){
                     return false;
              }
              @Override
              public void onCheckedChange(RadioGroup group, int
checkId){
                     //TODO Auto-generate method stub
                     switch(checkId){
```

```
                case R.Id.sateliteRB:
                                mp.setStreetView(false);
                                mp.setSatellite(true);
                                break;
                case R.Id.streetRB:
                                mp.setSatellite(false);
                                mp.setStreetView(true);
                                break;
                                }
                        }
                }
```

Sumber : [AHA-12]

2.3.2 *MapView*

MapView merupakan kelas berupa tampilan yang digunakan untuk menampilkan peta dunia digital, yang mana data-datanya didapatkan dari layanan *Google Maps*. Google Map API merupakan aplikasi *interface* yang dapat diakses dengan *javascript* agar *Google Maps* dapat ditampilkan pada halaman web yang dibangung. Untuk dapat mengakses *Google Maps*, harus melakukan pendaftaran *Api Key* terlebih dahulu dengan data pendaftaran berupa nama domain web yang dibangun. Tampilan ini dapat menangkap *touch gesture* sehingga dapat dilakukan penggeseran dan perbesaran pada peta tersebut. Tampilan peta dapatdiubah-ubah ke dalam tiga mode tampilan, yaitu tampilan *Street View*, *Satellite View*, dan *Traffic View*. Pada tabel 2.3 merupakan contoh kode program *MapView* dan pada gambar 2.2 merupakan contoh tampilan gambar *Google Maps.*

Tabel 2.3 Contoh kode program *MapView*

```
private void setupMapIfNeeded()
{
if (map == null)        {
FragmentManager                 fragmentManager                 =
getSupportFragmentManager();
 SupportMapFragment supportMapFragment = (SupportMapFragment)
        fragmentManager.findFragmentById(R.id.maps);
map = supportMapFragment.getMap();
```

```
if (map != null)
{
setupMap();
}
}
}
```

Gambar 2.2 Tampilan *Google Maps*

Sumber : [*Google Maps*]

2.3.3 OverlayItem

Kelas ini merupakan kelas abstrak yang digunakan sebagai kelas dasar bagi sebuah *Overlay* yang terdiri atas daftar-daftar *OverlayItems*. Sesuai dengan namanya, kelas ini memungkinkan untuk mendaftarkan *point of interest* yang akan ditampilkan di peta [MML-09:05]. Kelas ini menangani beberapa hal seperti proses penggambaran penanda (marker) untuk setiap titik dan menangani pula proses untuk menindak lanjuti adanya tap yang dilakukan pengguna terhadap item. Pada tabel 2.4 merupakan contoh kode program *OverlayItem.*

Tabel 2.4 Contoh kode program *OverlayItem*

```
public kelas CustomItemizedOverlay extends
ItemizedOverlay<OverlayItem> {
```

```
	private ArrayList<OverlayItem> mapOverlays = new
ArrayList<OverlayItem>();
	private Context context;
	public CustomItemizedOverlay(Drawable defaultMarker) {
		super(boundCenterBottom(defaultMarker));
		//TODO Auto-generated coonstructor stub
		}
	public CustomItemizedOverlay(Drawable defaultMarker, Context
context){
		this(defaultMarker);
		this.context = context;
		}
		@Override
	protected OverlayItem createItem(int i){
			//TODO Auto-generated method stub
			return mapOverlays.get(i);
			}
			@Override
		public int size(){
			//TODO Auto-generated method stub
			}
		@Override
	protected boolean onTap(int index){
	OverlayItem item = mapOverlays.get(index);
	AlertDialog.Builder ad = new AlertDialog.Builder(context);
	ad.setMessage(Item.getSnippet());
	ad.show();
	return true;
	}
	public void addOverlay(OverlayItem overlay){
		mapOverlays.add(overlay);
		this.populate();
		}
	}
```

Sumber : [AHA-12]

Gambar 2.3 merupakan contoh gambar dari *OverlayItem*, yaitu contoh gambar untuk penanda (marker) pada peta.

Gambar 2.3 ViewMarker

Sumber :[JKM-13]

2.3.4 *GeoPoint*

Kelas *GeoPoint* merupakan kelas yang merepresentasikan sepasang titik lokasi geografis yaitu lintang dan bujur. Titik tersebut disimpan sebagai angka integer dalam satuan mikroderajat. Konstruktor kelas ini adalah GeoPoint(int latitudeE6, int longitudeE6). *GeoPoint* juga perangkat lunak yang ideal untuk pemetaan geografis, mampu menyimpan lintang, bujur, ketinggian dan presisi, memperbarui tanda dalam *real time* pada peta agar mudah dilihat. Pada tabel 2.5 merupakan contoh program *GeoPoint.*

Tabel 2.5 Contoh kode program *GeoPoint*

```
        private void getGeoPointPengguna(double lat, double lon){
        //TODO Auto-generated method stub
        GeoPoint point = new GeoPoint((int)(lat * 1E6),(int)(lon *
1E6));
        OverlayItem overlayitem = new OverlayItem(point,"haii..",
"saya omayib")'
        itemizedOverlay = new
        CustomItemizedOverlay(this.getResources()
                .getDrawable(R.drawable.marker),
        MapMarker.this);
                itemizedOverlay.addOverlay(overlayitem);
                mapOverlays.add(itemizedOverlay);
```

```
            mapController.animateTo(point);
            mapController.seZoom(6);
        }
```

Sumber [AHA-12]

Gambar 2.4 merupakan contoh gambar tampilan *Google Maps* pada *smartphone* Android.

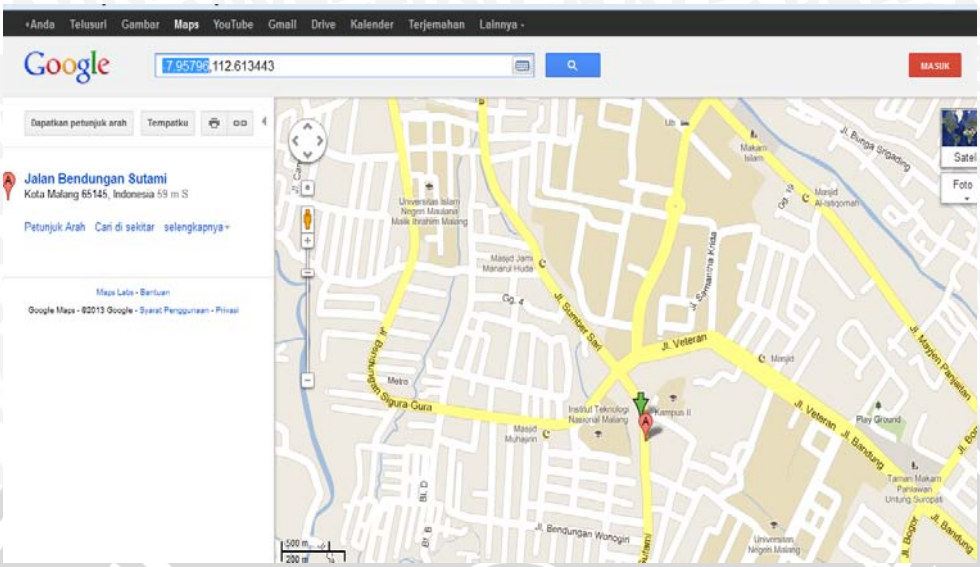
Gambar 2.4 Tampilan Google Maps pada smartphone android

Sumber :[JKM-13:03]

2.4 Android Location API

2.4.1 Geocoder

Kelas *Geocoder* merupakan kelas yang dipergunakan untuk menangani *geocoding* dan *reverse geocoding* [JKM-13:03]. *Geocoding* sendiri merupakan proses konversi dari data geografis (misalnya alamat atau kode pos) menjadi koordinat geografis dalam bentuk lintang dan bujur (misalnya -7.057006, 110.432798) dan hasil angka tersebut digunakan untuk penentuan titik lokasi-lokasi yang dibutuhkan oleh pengguna untuk mencari suatu tempat tertentu. Sementara *Reverse Geocoding* sendiri merupakan kebalikan dari *Geocoding*. Pada gambar 2.5 merupakan contoh gambar marker suatu lokasi pada *maps*.



Gambar 2.5 Contoh marker suatu lokasi

Sumber : [*Google Maps*]

2.4.2 Location

Kelas *Location* merupakan kelas yang merepresentasikan suatu lokasi geografis tertentu pada map. Dalam kelas ini terdapat method-method yang berguna yang berhubungan dengan suatu lokasi. Untuk mencari jarak tertentu dapat digunakan method *distanceTo (Location dest)* dengan memberikan parameter lokasi tujuan tertentu yang diinginkan.

2.4.3 *Location Manager*

Kelas ini memberikan akses pada layanan lokasi sistem. Hal ini memungkinkan aplikasi untuk mendapatkan pembaharuan dari lokasi geografis perangkat Android. Pada kelas ini, terdapat atribut-atribut yang berguna dalam proses pembaharuan lokasi geografis perangkat. Dua di antaranya adalah *GPS_PROVIDER dan NETWORK_PROVIDER*. Kedua atribut tersebut bertipe String.

2.4.4 *Location Listener*

LocationListener merupakan antarmuka yang digunakan untuk menerima pemberitahuan dari *Location Manager* ketika lokasi perangkat Android berubah [WNE-12:02]. *Location Listener* dipanggil dari method *request Location*

Updates(*String provider*, *long minTime*, *float minDistance*, *Location Listener listener*) pada *Location Manager*. Pada tabel 2.6 merupakan contoh kode program dari *location*, *locationmanager*, dan *locationListener* :

Tabel 2.6 Contoh kode program *Location*

```
public void onCreate(Bundle savedInstanceState){
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);

      mapView = (MapView) findViewById(R.id.map_view);
      mapView.setBuiltInZoomControls(true);
      mapOverlays = mapView.getController();
      pos = new penggunaPosition();

      LocationManager mLOcationManager = (LocationManager)
      getSystemService(context.LOCATION_SERVICE);
      locationListener = new MyLOcationListener();
      mLocationManager.requestLocationUpdate
      (LocationManager.GPS_PROVIDER,0,0, locationListener);

      if (pos.getLatitude() > 0){
      getGeoPointPengguna(pos.getLatitude(),
      pos.getLongitude());

      }else{
      getGeoPointPengguna(-7.801037, 110.364756);
      }
                              }
```

Sumber : [AHA-12]

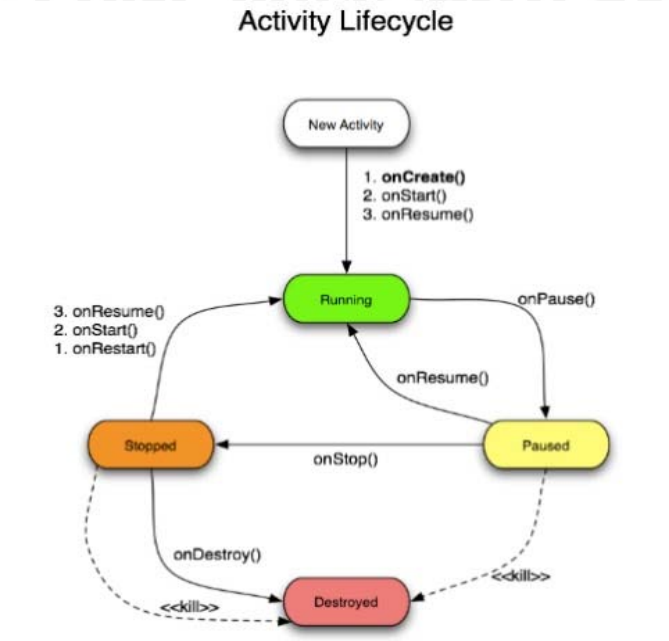
Gambar 2.6 dibawah ini merupakan contoh gambar suatu lokasi pengguna pada peta (maps). Terlihat titik biru pada peta menunjukkan lokasi seseorang / pengguna ketika berada disuatu tempat.

Gambar 2.6 *Location*

Sumber : [*Google Maps Location*]

2.5 Android

Android merupakan sistem operasi yang berbasiskan Linux yang penggunaannya ditujukan untuk perangkat bergerak. Pada awalnya Android dikembangkan oleh sebuah perusahaan bernama Android Inc [WNE-12:10]. Saat itu *Google Inc* sudah memprediksi bahwa perangkat bergerak akan berkembang dengan pesat. Akhirnya Google mengakuisisi *Android Inc* pada tahun 2005. Setelahnya, Android dikembangkan oleh *Google Inc* dengan sifat open source. Google menyediakan kemudahan untuk pengembang perangkat lunak dengan adanya *plugin Android Development Tool (ADT), Android Software Development Kit (SDK)* untuk Eclipse dengan bahasa pemrograman Java. Android tidak menggunakan *Java Virtual Machine (JVM)* seperti aplikasi Java pada umumnya. Android mempunyai Virtual Machine sendiri yang disebut Dalvik Virtual Machine yang merupakan software stack. Pada gambar 2.7 merupakan contoh siklus hidup aplikasi Android.



Gambar 2.7 Siklus hidup aplikasi Android

Sumber : [AHA-12]

Selama siklus ini berjalan, *activity* bisa mempunyai lebih dari 2 status seperti yang terlihat pada gambar 2.7. Kita tidak bisa mengontrol setiap status karena semua sudah ditangani oleh sistem. Namun kita akan mendapat pesan saat terjadi perubahan status melalui method *onXX()* [AHA-12]. Berikut adalah penjelasan untuk setiap status:

a. *onCreate(Bundle)* : Dipanggil saat pertama kali aplikasi dijalankan. Kita dapat menggunakan ini untuk deklarasi variabel atau membuat pengguna *interface.*

b. *onStart()* : Mengindikasikan Activity yang ditampilkan ke pengguna (pengguna).

c. *onResume()* : Dipanggil saat aplikasi kita mulai berinteraksi dengan pengguna. Di sini sangat cocok untuk meletakkan animasi ataupun musik.

d. *onPause()* : Dipanggil saat aplikasi yang kita jalankan kembali ke halaman sebelumnya atau biasanya karena ada *Activity* baru yang dijalankan. Di sini cocok untuk meletakkan algoritma penyimpanan (*save*).

e. *onStop()* : Dipanggil saat aplikasi kita berjalan dibelakang layar dalam waktu cukup lama.

f. *onRestart()* : *Activity* kembali menampilkan pengguna *interface* setelah status stop.

g. *onDestroy()* : Dipanggil saat aplikasi benar-benar berhenti.

h. *onSaveInstanceState()* : Method ini mengizinkan *activity* untuk menyimpan setiap status *instance*. Sebagai contoh dalam mengedit teks, kursor bergerak dari kiri ke kanan.

i. *onRestoreInstanceState(Bundle)*: Dipanggil saat *activity* kembali menginisialisasi dari status sebelumnya yang disimpan oleh *onSaveInstanceState (Bundle)*.

2.6 Mengenal *JSON*

JSON adalah *JSON (JavaScript Object Notation)* adalah format pertukaran data *(lightweight data-interchange format)*, mudah dibaca dan ditulis oleh manusia, serta mudah diterjemahkan dan dibuat *(generate)* oleh komputer. Format ini dibuat berdasarkan bagian dari Bahasa Pemprograman JavaScript, Standar ECMA-262 Edisi ke-3 – Desember 1999. *JSON* merupakan format teks yang tidak bergantung pada bahasa pemprograman apapun karena menggunakan gaya bahasa yang umum digunakan oleh programmer keluarga C termasuk C, C++, C#, *Java*, *JavaScript*, *Perl*, *Python* dll. Oleh karena sifat-sifat tersebut, menjadikan

JSON ideal sebagai bahasa pertukaran-data. Pada tabel 2.7 merupakan contoh kode program *JSON*.

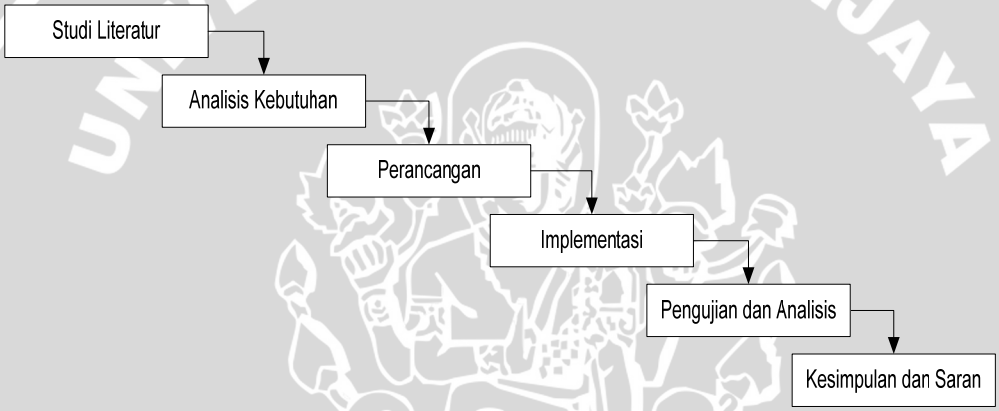
Tabel 2.7 Contoh kode program *JSON*

```
Public class JSONHelper
{
     private InputStream      is                = null;
      private JSONObject      jsonObject        = null;
      private String          json              = "";
      private final String    TAG_TambalBan     = "tambalban";
     private final String     TAG_ID            = "id";
      private final String    TAG_NAMA          = "nama";
      private final String    TAG_ALAMAT        = "alamat";
      private final String    TAG_LAT           = "lat";
      private final String    TAG_LNG           = "lng";
      public JSONObject getJSONFromURL(String url)
      {
      try
      {
      DefaultHttpClient httpClient = new DefaultHttpClient();
      HttpGet httpGet = new HttpGet(url);
      HttpResponse httpResponse = httpClient.execute(httpGet);
      HttpEntity httpEntity = httpResponse.getEntity();
      is = httpEntity.getContent();
      } catch (UnsupportedEncodingException e)
      {
      e.printStackTrace();
      } catch (ClientProtocolException e)
      {
      e.printStackTrace();
      } catch (IOException e)
      {
      e.printStackTrace();
      }
      try
      {
BufferedReader reader = new BufferedReader(new InputStreamReader(
            is, "iso-8859-1"), 8);
```

```
        StringBuilder sb = new StringBuilder();
        String line = null;
        while ((line = reader.readLine()) != null)
        {
        sb.append(line + "\n");
        }
        is.close();
        json = sb.toString();
        } catch (Exception e)
        {
              // TODO: handle exception
        }
        try
        {
              jsonObject = new JSONObject(json);
        } catch (JSONException e)
        {
              // TODO: handle exception
        }
  return jsonObject;
  }
```

BAB III
METODOLOGI PENELITIAN

Pada bab ini dijelaskan langkah-langkah yang akan dilakukan dalam pengerjaan skripsi, yaitu studi literatur dan penyusunan dasar teori, analisa dan perancangan, implementasi, pengujian dari aplikasi perangkat lunak yang akan dibuat, hingga penulisan laporan. Kesimpulan dan saran disertakan sebagai catatan atas aplikasi dan kemungkinan arah pengembangan aplikasi selanjutnya.Gambar 3.1 menunjukkan tahapan penelitian secara umum.



Gambar 3.1 Tahapan penelitian

3.1 Studi Literatur

Studi literatur menjelaskan dasar teori yang digunakan untuk menunjang penulisan skripsi. Teori-teori pendukung tersebut yaitu :

1. Kajian pustaka
2. *Global Positioning Sistem (GPS)*
3. *Google Maps API*
4. *Android Location API*
5. Android
6. *JSON*

3.2 Analisis Kebutuhan Sistem

Analisis kebutuhan digunakan untuk mendapatkan kebutuhan aplikasi bengkel yang akan dibangun. Metode analisis yang digunakan adalah *object-oriented analysis* dengan menggunakan bahasa pemodelan UML *(Unified Modeling Language).* Diagram *use case* digunakan untuk mendeskripsikan kebutuhan-kebutuhan dan *fungsionalitas* perangkat lunak dari sudut pandang pengguna. Analisis kebutuhan dilakukan dengan mengidentifikasi semua kebutuhan aplikasi bengkel yang kemudian akan dimodelkan dalam diagram *use case.*

3.3 Perancangan

Perancangan perangkat lunak dilakukan setelah semua kebutuhan perangkat lunak didapatkan melalui tahap analisis kebutuhan. Perancangan perangkat lunak berdasarkan *object-oriented analysis* dan *object-oriented design* yaitu menggunakan pemodelan UML (*Unified Modeling Language*). Perancangan dimulai dari perancangan alur atau aktifitas yang dilakukan pengguna secara prosedural yang dimodelkan dalam *activity diagram.* Interaksi antar objek yang telah diidentifikasi, dimodelkan dalam *sequence diagram*. Selanjutnya, dilakukan perancangan sistem aplikasi bengkel dengan mengidentifikasi *class* dan *layout* yang dibutuhkan, serta kemudian dimodelkan dalam *class diagram*. Kemudian tahap perancangan dilanjutkan dengan perancangan antarmuka pengguna.

3.4 Implementasi

Implementasi perangkat lunak mengacu kepada perancangan perangkat lunak. Implementasi perangkat lunak diawali dengan penjabaran spesifikasi lingkungan perancangan perangkat lunak. Selanjutnya dijabarkan *mapping class* dengan *layout* saat implementasi perangkat lunak. Implementasi perangkat lunak dilakukan dengan menggunakan bahasa pemrograman Java. Implementasi antarmuka berdasarkan perancangan yang telah dilakukan. Pada tahap akhir dilakukan implementasi pada *smartphone* Android secara langsung. Pemasangan aplikasi pada *smartphone* menggunakan ADB *(Android DebugBridge).*

3.5 Pengujian dan Analisis

Pengujian perangkat lunak dilakukan untuk mengetahui apakah kinerja dan performa sistem aplikasi bengkel telah sesuai dengan spesifikasi kebutuhan yang melandasinya. Terdapat dua macam pengujian yang dilakukan pada aplikasi ini yaitu pengujian *validasi,* dan pengujian *usability*. Pengujian validasi menggunakan metode *black-box testing*, sedangkan pengujian *usability* dilakukan untuk mengetahui tingkat kemudahan penggunaan sistem dengan pemberian kuisioner kepada pengguna. Setelah tahap pengujian, dilakukan analisis untuk mengetahui hasil dari pengujian perangkat lunak sehingga dapat didapatkan kesimpulan dari rancang bangun aplikasi yang telah dibuat.

3.6 Pengambilan Kesimpulan dan Saran

Pengambilan kesimpulan dilakukan setelah semua tahapan perancangan perangkat lunak, implementasi perangkat lunak, dan pengujian perangkat lunak telah selesai dilakukan. Kesimpulan diambil dari hasil pengujian dan analisis terhadap sistem yang dibangun. Tahap terakhir dari penulisan adalah saran yang dimaksudkan untuk memperbaiki kesalahan-kesalahan yang terjadi dan menyempurnakan penulisan serta untuk memberikan pertimbangan atas pengembangan perangkat lunak lebih lanjut.

BAB IV
ANALISIS DAN PERANCANGAN

Pada bab ini menjelaskan tentang analisis sistem dan desain dari aplikasi bengkel pada *smartphone* Android. Analisis sistem dan desain sistem memiliki dua tahap. Tahap pertama adalah proses analisis kebutuhan dilanjutkan dengan tahap kedua yaitu proses perancangan perangkat lunak. Tahap analisis kebutuhan terdiri atas lima langkah yaitu melakukan penjabaran gambaran umum aplikasi bengkel, identifikasi lokasi bengkel, penjabaran tentang daftar kebutuhan, kemudian memodelkannya ke dalam diagram *use case* dan diagram *activity*. Proses perancangan perangkat lunak memiliki lima tahap, yaitu perancangan arsitektural, perancangan basis data, pemodelan diagram *sequence* untuk menggambarkan interaksi antar objek di dalam aplikasi bengkel, pemodelan diagram *class* untuk menggambarkan perancangan struktur *class-class* yang menyusun aplikasi bengkel, dan perancangan antarmuka pengguna.

4.1 Analisis Kebutuhan Sistem

Untuk analisis kebutuhan ini, penulis melakukan survei langsung dan melakukan wawancara kepada setiap pengguna motor yang mengalami kerusakan kendaraan ataupun ban bocor, dan hal pertama yang disebutkan adalah mencari bengkel atau tambal ban terdekat dengan keberadaannya. Tetapi pengguna masih kesulitan ketika dijumpai dengan persimpangan jalan, dan tidak tahu harus pergi ke arah yang mana untuk mendapatkan lokasi bengkel yang lebih dekat. Maka penulis merasa perlu adanya sistem pencarian yang memudahkan pengguna untuk mendapatkan lokasi bengkel yang terdekat dengan keberadaannya saat itu.

Proses analisis kebutuhan ini diawali dengan penjabaran gambaran umum aplikasi bengkel, identifikasi lokasi bengkel, penjabaran tentang daftar kebutuhan dan kemudian dimodelkan kedalam diagram *use case*. Analisis kebutuhan ini bertujuan untuk menggambarkan kebutuhan-kebutuhan yang harus disediakan oleh sistem agar dapat memenuhi kebutuhan pengguna.

4.1.1 Gambaran Umum Aplikasi Bengkel

Aplikasi bengkel ini merupakan aplikasi yang dirancang untuk mempermudah pengguna untuk mendapatkan lokasi bengkel terdekat dengan keberadaan pengguna saat itu. Ketika mengakses aplikasi bengkel, pengguna hanya memerlukan waktu kurang lebih sepuluh detik, maka pengguna sudah mendapatkan lokasi bengkel terdekat dengan keberadaanya.

4.1.2 Identifikasi data lokasi Bengkel

Data lokasi bengkel pada aplikasi ini diperoleh dengan cara melakukan survei kesetiap lokasi bengkel yang ada di kota Pamekasan dan kota Malang. Survei yang dilakukan meliputi pendataan alamat, koordinat dan nama bengkel dengan cara mengunjungi satu persatu lokasi bengkel yang ada. Selanjutnya koordinat yang didapatkan digunakan untuk menentukan lokasi bengkel didalam map.

4.1.3 Daftar Kebutuhan

Daftar kebutuhan terdiri dari kebutuhan fungsional dan non-fungsional. Pada tabel 4.1 kebuthan fungsional ditunjukkan dengan penomoran F, sedangkan kebutuhan non-fungsional ditunjukkan dengan penomoran NF.

Tabel 4.1 Tabel kebutuhan fungsional

Identifikasi	Kebutuhan	Use case
F01	Aplikasi harus bisa menampilkan lokasi bengkel terdekat dengan keberadaan pengguna	Melihat lokasi bengkel
F02	Aplikasi harus bisa menampilkan informasi bengkel sesuai keinginan pengguna	Melihat informasi bengkel
F03	Aplikasi harus bisa menampilkan rute menuju bengkel yang ditentukan pengguna	Melihat rute lokasi bengkel
F04	Aplikasi harus bisa mengelola (menambahkan, mengubahdan menghapus) data lokasi bengkel	Manipulasi data

Tabel 4.1 merupakan tabel kebutuhan fungsional yang terdiri empat kebutuhan sistem. Masing-masing kebutuhan sistem harus dapat dipenuhi oleh

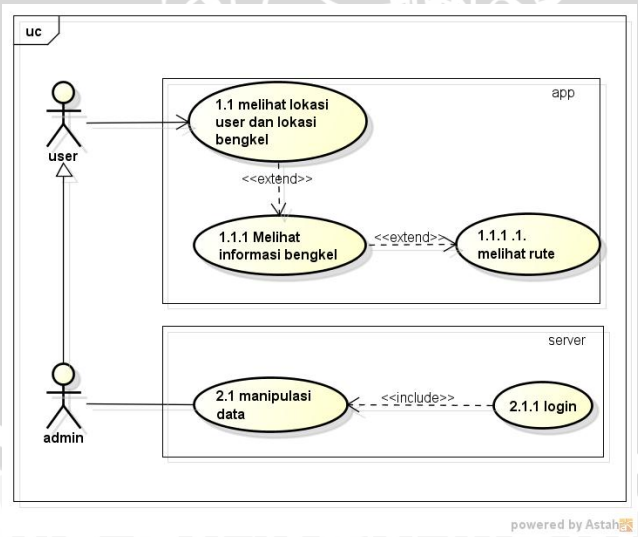
sistem. Setiap kebutuhan sistem dimodelkan ke dalam bentuk *use case*, dan setiap kebutuhan fungsional harus dilakukan pengujian terlebih dahulu sebelum sistem dipublikasikan. Tabel 4.2 merupakan daftar kebutuhan non-fungsional.

Tabel 4.2 Daftar kebutuhan non-fungsianal

Parameter	Kode	Deskripsi Kebutuhan
Usability	NF01	Aplikasi harus dapat dengan mudah digunakan oleh pengguna (Ketika pengguna menekan *icon* aplikasi bengkel yang telah terpasang pada *smartphone* Android, maka sistem akan langsung menampilkan keberadaan pengguna dan marker lokasi bengkel terdekat dengan keberadaan pengguna saat itu)
Compatibility	NF02	Aplikasi harus dapat digunakan pada *smartphone* dengan sistem operasi Android
Security	NF04	Aplikasi harus dapat mengidentifikasi pengguna yang dapat mengelola data seperti yang dituliskan pada kebutuhan F04

4.1.4 Diagram *Use Case*

Pemodelan *use case* sistem diperoleh dari kebutuhan fungsional yang digunakan untuk menggambarkan interaksi antara satu atau lebih aktor dengan sistem yang akan dibuat. Gambar 4.1 menunjukkan use case sistem bengkel.



Gambar 4.1 *Use Case* sistem bengkel

Gambar 4.1 merupakan *use case* sistem bengkel yang terdiri dari 2 aktor yaitu pengguna dan admin, dimana admin dapat memanipulasi data lokasi bengkel dan

mengakses aplikasi bengkel, sedangkan penggunahanya dapat melakukan akses aplikasi bengkel dan tidak dapat memanipulasi data lokasi bengkel. Setiap *use case* yang ada pada Gambar 4.1 dapat dijelaskan oleh skenario *use case*.

4.1.5 Skenario *Use Case*

Tabel 4.3 Skenario *Use Case* Melihat Lokasi Bengkel

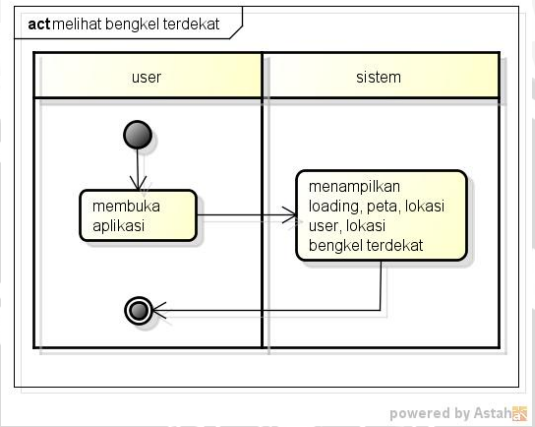
Nomor *Use Case*	F01
Nama *Use Case*	Melihat lokasi bengkel
Prasyarat Konteks	Aplikasi telah terpasang pada *smartphone* Android pengguna
Tujuan Dalam Konteks	Mempermudah pengguna untuk menentukan lokasi bengkel terdekat
Prakondisi	Aplikasi sudah terpasang dan pengguna membuka aplikasi bengkel
Kondisi Akhir Sukses	Aplikasi menampilkan lokasi pengguna dan menampilkan bengkel terdekat dengan keberadaan pengguna
Kondisi Akhir Failed	-
Aktor	Pengguna
Alur Utama	Aktifitas
	1. Pengguna membuka aplikasi bengkel 2. Sistem menampilkan *loading*, peta, lokasi pengguna dan lokasi bengkel terdekat dengan keberadaan pengguna

Tabel 4.3 menjelaskan kegiatan yang dilakukan pada saat melihat lokasi bengkel pada gambar 4.2 *use case* sistem bengkel. Pertama pengguna menekan *icon* dari aplikasi yang telah terpasang pada *smartphone* Android, kemudian sistem akan menampilkan *loading* (untuk mengambil data lokasi bengkel pada *database* dan ditampilkan pada peta), menampilkan peta, marker lokasi bengkel, dan lokasi pengguna pada saat itu. Tabel Skenario *use case* untuk penjelasan tiap *use case* yang selanjutnya dapat dilihat pada lampiran 2 dan 3.

4.1.6 Diagram Activity

Diagram *activity* adalah diagram untuk memodelkan aktivitas antara pengguna dan sistem yang berjalan berdasarkan pada scenario *use case*. Skenario

use case yang digambarkan pada tabel 4.3 dapat dimodelkan kedalam *diagram activity* pada gambar 4.2.dengan *diagram activity*, dapat terlihat aktor yang melakukan proses tiap langkahnya.



Gambar 4.2 Aktivitas diagram melihat lokasi bengkel

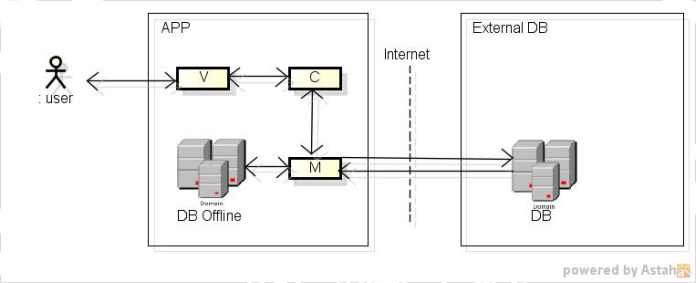
Gambar 4.2 merupakan aktivitas diagram melihat lokasi bengkel. Aktivitas diagram dibuat untuk menjelaskan interaksi antara pengguna dengan sistem. Pertama penggunamenekan *icon* aplikasi bengkel, kemudian sistem menampilkan *loading* (untuk mengambil data lokasi bengkel pada *database* dan ditampilkan pada peta), menampilkan peta, menampilkan lokasi pengguna saat itu dan menampilkan lokasi bengkel terdekat. Untuk penjelasan tiap *Aktivitas* diagram yang selanjutnya dapat dilihat pada lampiran 5 dan 6.

4.2 Perancangan Perangkat Lunak

Perancangan aplikasi bengkel dimulai dari perancangan arsitektural, perancangan basis data, pemodelan diagram *sequence* untuk menggambarkan interaksi antar objek di dalam aplikasi bengkel. Pemodelan diagram *class* untuk menggambarkan perancangan struktur *class* yang menyusun aplikasi bengkel, dan perancangan antarmuka pengguna. Perancangan perangkat lunak pada penelitian ini menggunakan pendekatan desain berorientasi objek yang direpresentasikan dengan menggunakan UML (*Unified Modelling Language*).

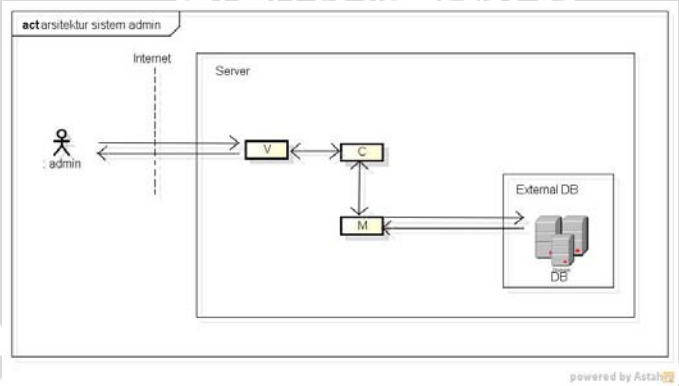
4.2.1 Perancangan Arsitektural

Perancangan arsitektural pada aplikasi bengkel menggunakan metode *MVC (Model View Controller)*. Dalam Gambar 4.3 *view* merupakan tampilan antar muka yang digunakan untuk berinteraksi dengan sistem. *Controller* merupakan jembatan yang menghubungkan antara *view* dan model. Pada aplikasi bengkel, model berisi tentang alamat bengkel, lokasi dan nama bengkel.



Gambar 4.3 Arsitektur sistem pengguna

Dalam Gambar 4.3 menjelaskan tentang arsitektur sistem pengguna ketika melihat lokasi bengkel. Terdapat dua cara untuk pengaksesan data yakni *online* dan *offline*. Perbedaannya terdapat pada saat pengaksesan model, model akan mengakses *externalDB* jika pengguna tehubung dengan *internet*, kemudian data akan dikembalikan melalui *controller* untuk ditampilkan pada pengguna melalui *view*. Sedangkan dalam gambar 4.4 menjelaskan tentang arsitektur sistem admin.



Gambar 4.4 Arsitektur sistem admin

Dalam gambar 4.4 admin mengelola data di *eksternalDB* yang ada di dalam *server*. Admin melakukan koneksi *internet* dalam mengelola (tambah, hapus, edit) data lokasi bengkel yang akan digunakan untuk aplikasi bengkel.

4.2.2 Perancangan Basis Data

Perancangan basis data pada aplikasi bengkel hanya digunakan untuk menyimpan data lokasi bengkel yang telah direalisasikan. Data lokasi bengkel terdiri dari beberapa *field* diantaranya adalah (id, nama, longitude, latitude, status, jam_buka, jam_tutup dan alamat).

Id merupakan *field* yang digunakan untuk penomoran (angka) pada tiap-tiap lokasi bengkel, sedangkan nama digunakan untuk menampilkan nama tiap-tiap bengkel, *longitude* dan *latitude* digunakan untuk proses konversi dari data geografis (misalnya alamat atau kode pos) menjadi koordinat geografis dalam bentuk lintang dan bujur (misalnya -7.057006, 110.432798) dan hasil angka tersebut digunakan untuk penentuan titik lokasi-lokasi yang dibutuhkan oleh pengguna untuk mencari suatu tempat tertentu yang di sebut *Geocoding.* Status merupakan *field* untuk menampilkan status dari bengkel tersebut apakah masih beroperasi atau sudah tidak beroperasi lagi, jam_buka dan jam_tutup merupakan *field* yang menampilkan waktu bengkel tersebut mulai buka dan waktu tutup dan *field* alamat digunakan untuk menampilkan alamat-alamat di tiap-tiap lokasi bengkel.

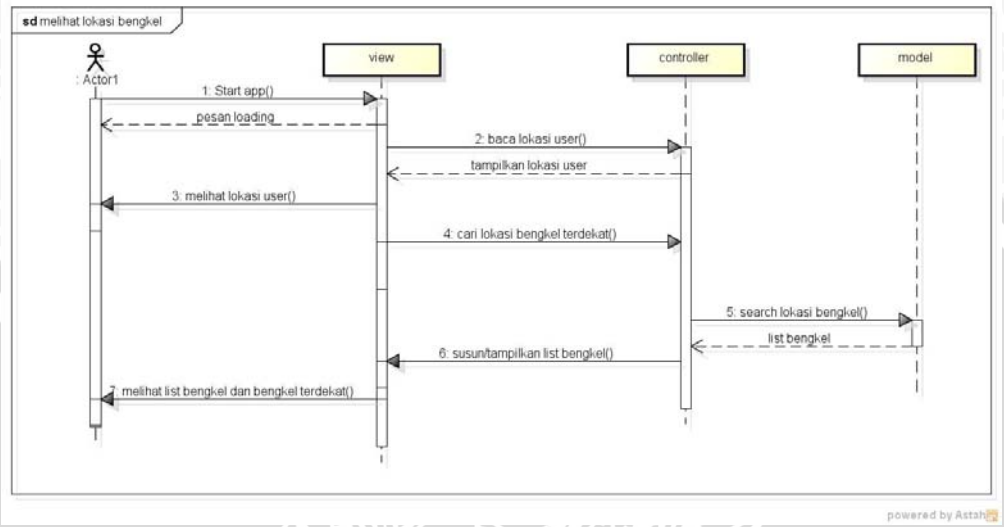
Masing-masing *field* tersebut mempunyai tipe data yang berbeda dan dikelompokkan dalam satu tabel. Pada aplikasi bengkel, admin dapat mengelola (menambahkan, mengubah dan menghapus) data lokasi bengkel. Pada tabel 4.4 merupakan rancang tabel untuk penyimpanan lokasi bengkel.

Tabel 4.4 Rancang tabel bengkel untuk menyimpan data lokasi bengkel

Tabel_Bengkel	
Field	**Type Data**
Id	Int
nama	Varchar
longitude	Double
latitude	Double
status	Varchar
jam_buka	Time
jam_tutup	Time
alamat	Varchar

4.2.3 Perancangan Diagram *Sequence*

Diagram *sequence* digunakan untuk menggambarkan interaksi antar objek di dalam aplikasi bengkel. Pada gambar 4.5 merupakan gambar diagram *sequence* untuk melihat lokasi bengkel.

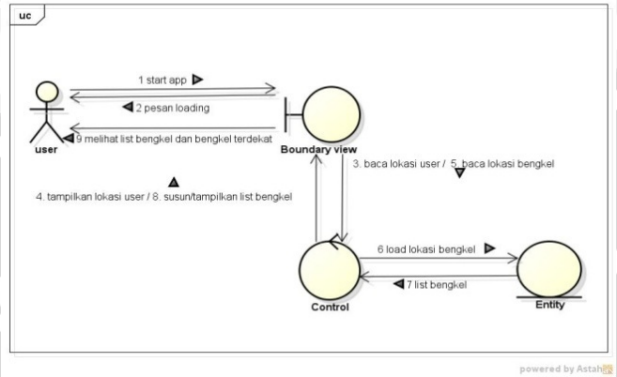


Gambar 4.5 Diagram *sequence* melihat lokasi bengkel

Dalam Gambar 4.5 dijelaskan pada saat pengguna menekan tombol *icon* aplikasi bengkel, sistem memulai aktivitas aplikasi kemudian menjalankan method *start application()* untuk menampilkan *map* pada *view*. Pesan *loading* berfungsi untuk mengetahui waktu yang diperlukan ketika sistem melakukan pengambilan data lokasi bengkel yang telah tersimpan di dalam *database*. Waktu pengambilan data dari *database* yang diperlukan kurang lebih 10 detik sebelum akhirnya selesai dan dilanjutkan dengan tampilan peta, posisi pengguna dan marker lokasi bengkel yang ada pada *map*. Diagram *sequence* untuk kegiatan yang lain dapat dilihat pada lampiran 8 dan 9.

Untuk melengkapi diagram *sequence* pada aplikasi bengkel maka dibuatlah suatu diagram kolaborasi yang digunakan sebagai tahap awal untuk membuat perancangan class diagram. Diagram kolaborasi antar class digambarkan dalam gambar 4.6. Pengguna melakukan *start* pada aplikasi, aplikasi merespon dan menampilkan pesan *loading*, kemudian sistem akan menampilkan lokasi pengguna. *Class control* akan mengatur jalannya sistem ke *class entity* untuk

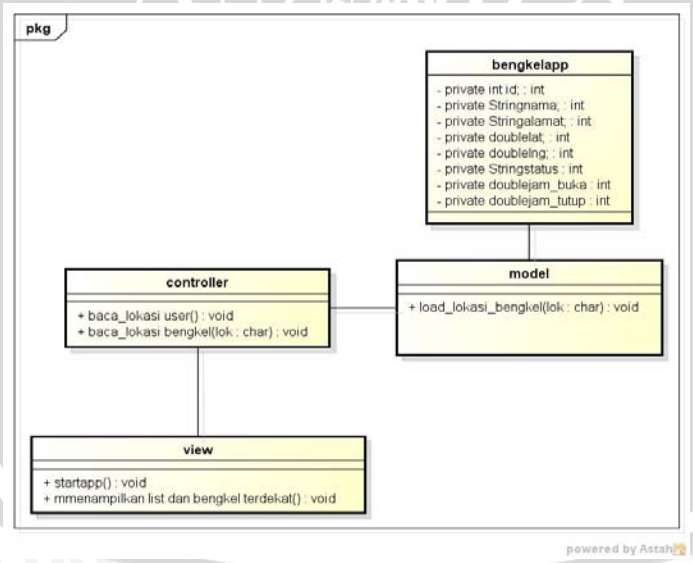
mencari lokasi bengkel dan mengembalikan data list bengkel ke *control* untuk ditampilkan pada boundary sehingga pengguna dapat melihat list bengkel serta bengkel terdekat. Pada gambar 4.6 merupakan gambar diagram kolaborasi antar objek.



Gambar 4.6 *Collaboration diagram* antar objek

4.2.4 Perancangan Diagram *Class*

Pemodelan diagram *class* yaitu, untuk menggambarkan perancangan struktur *class-class* yang menyusun aplikasi bengkel.



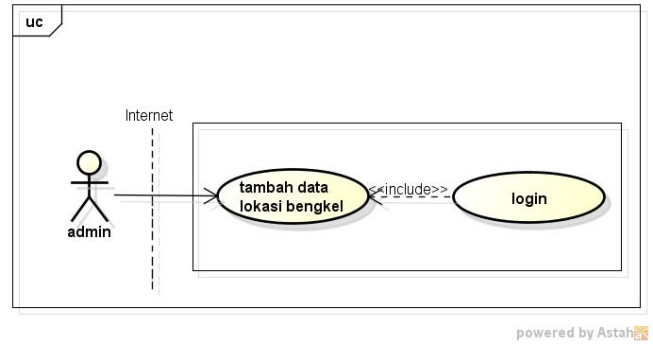
Gambar 4.7 *Class* diagram mmelihat lokasi bengkel

Dalam gambar 4.7 menjelaskan tentang *diagram class* aplikasi bengkel. Ketika pengguna melakukan klik *icon* aplikasi bengkel maka sistem akan menampilkan *startapp* pada *view* yang diteruskan oleh *control* dan *model* untuk

melanjutkan jalannya aplikasi. *Control* merupakan sebuah sistem yang melakukan *aktivitas* untuk pencarian lokasi pengguna dan lokasi bengkel kemudian ditampilkan pada *view*, sedangkan *model* melakukan *load data* (mengambil data) dari sebuah *database* yang merupakan penyimpanan lokasi-lokasi bengkel yang sudah ditentukan oleh admin sehingga nantinya akan ditampilkan pada peta.

4.2.5 Manipulasi data bengkel

Disetiap data bengkel yang ditampilkan pada aplikasi (ketika mode *online*) memerlukan seorang admin untuk mengolah data disetiap alurnya. Gambar 4.8 merupakan *use case* admin yang mengolah data bengkel pada *database*.



Gambar 4.8 *Use Case* manipulasi bengkel

Admin melakukan koneksi internet untuk bisa mengakses *database* (*server*) yang berisi data-data bengkel yang telah *direalisasikan.* Setelah melakukan *login* pada *database* (*server*), admin melakukan manipulasi data. Sebagai contoh, admin dapat melakukan tambah data lokasi bengkel pada *database* dengan menambahkan angka / letak geografis suatu lokasi bengkel tersebut yang telah ada pada *Google Maps*, kemudian menambahkan nama dan alamat bengkel tersebut. Setelah di lakukan *update* pada *database* tersebut oleh admin, maka pada aplikasi bengkel terlihat lokasi bengkel akan bertambah.

4.2.5.1 Skenario *Use Case*

Tabel 4.5 Skenario *Use Case* login

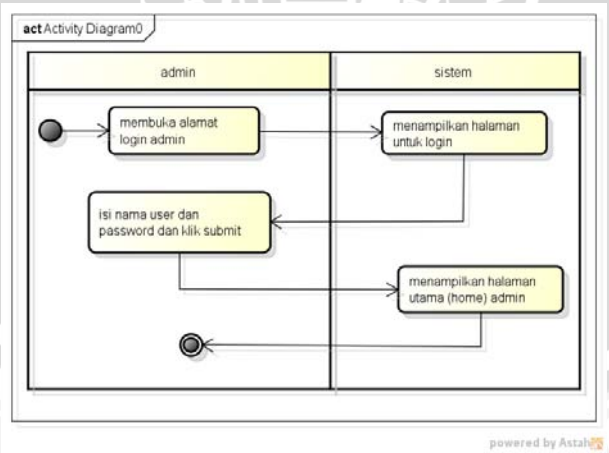
Nomor *Use Case*	F02
Nama *Use Case*	Login
Prasyarat Konteks	Admin sudah terkoneksi dengan internet (*localhost*)
Tujuan Dalam Konteks	Untuk memanipulasi data bengkel

Prakondisi	Admin sudah berhasil login
Kondisi Akhir Sukses	Aplikasi menampilkan halaman utama (*home*) admin setelah berhasil login
Kondisi Akhir Failed	-
Aktor	Admin
Alur Utama	Aktifitas
	1. Admin terkoneksi dengan internet (*localhost*) 2. Sistem menampilkanhalaman utama (*home*) admin

Tabel 4.5 menjelaskan kegiatan yang dilakukan admin pada saat *login* kehalaman admin. Pertama admin melakukan koneksi pada internet (*localhost*) dan melakukan *login* dengan memasukkan nama pengguna dan *password* untuk bisa masuk kedalam halaman *home* admin sehingga bisa memanipulasi data bengkel. untuk penjelasan tiap *use case* yang selanjutnya dapat dilihat pada lampiran 4.

4.2.5.2 Diagram *Activity*

Diagram *activity* adalah diagram untuk memodelkan *aktivitas* antara pengguna dan sistem yang berjalan berdasarkan pada scenario *use case.* Skenario *use case* yang digambarkan pada tabel 4.5 dapat dimodelkan kedalam diagram *activity* pada gambar 4.9. dengan diagram *activity*, dapat terlihat aktor yang melakukan proses tiap langkahnya.

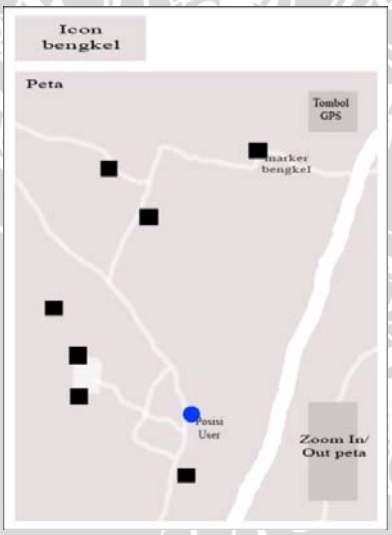


Gambar 4.9 *Aktivitas* diagram login halaman admin

Gambar 4.9 merupakan *aktivitas* diagram login halaman admin. *Aktivitas* diagram dibuat untuk menjelaskan interaksi antara admin dengan sistem. Pertama admin sudah terkoneksi dengan internet dan masuk ke alamat login admin, kemudian sistem menampilkan halaman untuk login. Setelah itu, admin melakukan login dengan mengisi *name* dan *password*, setelah berhasil login maka sistem akan menampilkan halaman utama (*home*). Untuk penjelasan tiap Aktivitas diagram yang selanjutnya dapat dilihat pada lampiran 7.

4.2.6 Perancangan Antarmuka

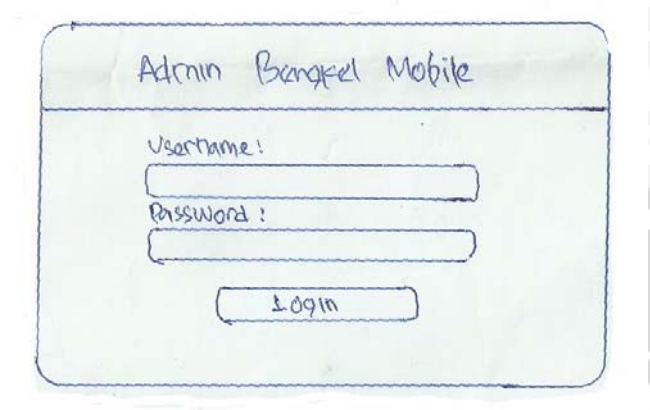
Perancangan antarmuka merupakan rancangan awal yang akan digunakan dalam implementasi antarmuka pengguna pada aplikasi bengkel. Perancangan tampilan antarmuka pada saat menampilkan peta, lokasi pengguna dan marker (lokasi bengkel) ditunjukkan dalam Gambar 4.10.



Gambar 4.10 Rancangan antarmuka aplikasi bengkel

Pada gambar 4.10 menjelaskan tentang tampilan awal aplikasi bengkel ketika dijalankan. Menampilkan peta, lokasi pengguna dan marker (lokasi bengkel). Untuk perancangan tampilan antarmuka pada saat melakukan klik *popup* (menampilkan informasi bengkel), mendapatkan rute bengkel dengan lokasi pengguna, dan melihat halaman about dapat dilihat pada lampiran 12 dan 13.

Sedangkan perancangan antarmuka admin pada saat *login* ditunjukkan pada gambar 4.11.



Gambar 4.11 Perancangan Admin Aaplikasi Bengkel

Gambar 4.11 merupakan tampilan perancangan *login* admin pada *database* untuk dapat mengolah data lokasi bengkel pada server.

BAB V
IMPLEMENTASI DAN PENGUJIAN SISTEM

Bab ini membahas mengenai tahapan implementasi dan pengujian aplikasi bengkel berdasarkan hasil yang telah didapatkan dari analisis kebutuhan dan perancangan aplikasi. Pembahasan terdiri dari implementasi basis data, implementasi *class*, dan implementasi antarmuka. Kemudian dilanjutkan dengan pengujian *validasi* dan pengujian *usability* pada aplikasi bengkel. Pada tahap terakhir dilakukan analisa untuk mendapatkan kesimpulan dari hasil pengujian aplikasi bengkel yang telah dilakukan. Proses analisis mengacu pada dasar teori sesuai dengan hasil pengujian yang didapatkan. Analisis dilakukan terhadap hasil pengujian di setiap tahap pengujian.

5.1 Implementasi Basis Data

Pada aplikasi bengkel menggunakan basis data sebagai tempat untuk menyimpan data lokasi-lokasi bengkel. Tabel 5.1 menjabarkan tentang analisis dari lokasi yang digunakan untuk menyimpan data lokasi bengkel.

Tabel 5.1 Tabel data bengkel

No.	Nama Field	Key	Type	Deskripsi
1	Id	PK	Int	penomoran data lokasi bengkel (1,2,3,..)
2	Nama		Varchar	nama bengkel (sinar jaya motor)
3	Longitude		Double	letak lokasi bengkel (113.485867)
4	Latitude		Double	letak lokasi bengkel (-7.160792)
5	Status		Varchar	bengkel sudah tutup atau masih beroprasi (buka/tutup)
6	jam_buka		Varchar	bengkel mulai beroprasi (07.00)

7	jam_tutup		Varchar	bengkel sudah tutup / tidak beroprasi (08.30)
8	Alamat		Varchar	alamat bengkel (Jl.Stadion no.28)

5.2 Implementasi *Class*

Setiap *class* yang telah dirancang pada proses perancangan direalisasikan pada sebuah *file* program dengan ekstensi *.java dan *file layout* dengan ekstensi *.xml sebagai tampilan dari *class* tersebut. *Class* dalam perancangan yang hanya berisi *method* tidak memiliki *layout* untuk menampilkan isinya. Tabel 5.2 menjelaskan mengenai pasangan antara *class* dengan *file* program dan *layout* yang digunakan untuk mengimplementasikannya.

Tabel 5.2 Implementasi *class* untuk program dan *layout*

No.	Package	Objek	Nama Class	Nama File Program	Nama File Layout
1	bengkel.app	View	MainActivity	MainActivity.Java	activity_main.xml
2	bengkel.app	Bengkelapp	Bengkelapp	Bengkelapp.Java	activity_info_bengkelapp.xml
3	bengkel.app	Control	Controller	MainActivity.Java	activity_main.xml
4	bengkel.app	Model	JsonHelper	JSONHelper.Java	activity_main.xml

Implementasi *class* dan *layout* pada tabel 5.2 dapat dihasilkan tampilan seperti pada gambar 5.1 pada tampilan antarmuka halaman utama aplikasi bengkel. Sedangkan tampilan antamuka yang lain dapat dilihat pada lampiran 10 dan 11.

5.3 Implementasi *Source Code Class MainActivity*

Perangkat lunak bengkel terdekat menggunakan *google maps* berbasis Android memiliki beberapa proses atau *method* yang terdapat pada beberapa *class.* Diantaranya *Class MainActivity, class* ini digunakan untuk menampilkan halaman utama dalam perangkat lunak bengkel berbasis Android. Di dalam *class* ini dijelaskan proses pengambilan data peta dari *Google Maps Api*, cek kondisi

ketika perangkat lunak mendapat koneksi internet atau tidak. *Source code* berikut menjelaskan proses yang terdapat di *class MainActivity*.

Tabel 5.3 Implementasi *Source code Class MainActivity myLocation*

Source code MainActyvity
private void moveToMyLocation() { LocationManager locationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE); Criteria criteria = new Criteria(); Location location = locationManager.getLastKnownLocation(locationManager.getBestProvider(criteria, false));

Pada tabel 5.3 menjelaskan tentang *source code Class MainActivity* bagaimana sistem *request* / meminta lokasi pengguna lewat *GPS* untuk menampilkan lokasi keberadaan pengguna pada peta.

Tabel 5.4 Implementasi *Source code Class MainActivity setMarker*

Source code MainActyvity

public void setMarker(){
map.addMarker(new MarkerOptions()
.position(new LatLng(-7.162277, 113.482504))
.title("Tambal Ban, Bengkel, & Variasi")
.snippet("Trunojoyo"));
 	
map.addMarker(new MarkerOptions()
.position(new LatLng(-7.159466, 113.483384))
.title("Tambal Ban")
.snippet("(Arek Lancor"));
}

Pada tabel 5.4 menjelaskan tentang *source code Class MainActivity* bagaimana memberikan marker / tanda suatu lokasi lokasi pada peta.

Tabel 5.5 Implementasi *Source code Class MainActivity current location*

Source code MainActyvity
myLocation = new LatLng(map.getMyLocation().getLatitude(), map.getMyLocation().getLongitude());

Pada tabel 5.5 menjelaskan tentang *source code Class MainActivity* untuk mengambil *latitude* dan *longitude* lokasi pengguna untuk dikirim ketika meminta ke *google direction.*

Tabel 5.6 Implementasi *Source code Class MainActivity Json*

Source code MainActyvity

`json.getJSONFromURL(URL_API);`
`listBengkelapp = json.getBengkelappAll(jObject);`
`return null;`

Pada tabel 5.6 menjelaskan tentang *source code Class MainActivity* bagaimana memanggil / *load* data lokasi bengkel yang nantinya akan ditampilkan pada peta di aplikasi bengkel

Tabel 5.7 Implementasi Source code Class MainActivity locationB

Source code MainActyvity

`locationB.setLatitude(listBengkelapp.get(i).getLat());`
`locationB.setLongitude(listBengkelapp.get(i).getLng());`
`distanceTemp  = locationA.distanceTo(locationB);`
`if(distanceTemp < distance){`
`distance = distanceTemp;`
`idxDist = i;`
`}`
`}`

Pada tabel 5.7 menjelaskan tentang *distanceTemp = locationA.distanceTO (locationB)* yaitu tentang perpindahan jarak lokasi A ke lokasi B. dimana lokasi A itu lokasi pengguna dan B lokasi bengkel / marker lokasi bengkel yang ada pada peta. Dapat disimpulkan bahwa *source code* pada tabel 5.7 menjelaskan tentang jarak terdekat ke lokasi bengkel / marker pada peta. Sedangkan pada tabel 5.8 merupakan Implementasi *Source code Class MainActivity popup* / informasi bengkel.

Tabel 5.8 Implementasi *Source code Class MainActivity locationB*

Source code MainActyvity

`if(i == idxDist){`
`map.addMarker(new MarkerOptions()`
`.position(new LatLng(listBengkelapp.get(i).getLat(), listBengkelapp.get(i).getLng()))`
`.title(listBengkelapp.get(i).getNama())`
`.snippet(listBengkelapp.get(i).getAlamat()).icon(BitmapDescriptorFactory.fromResource(R.drawable.tires2)));`
`}`
`else{`
`map.addMarker(new MarkerOptions()`
`.position(new MarkerOPtion()`

Pada tabel 5.8 menjelaskan *source code MainActivity* bagaimana menentukan dan menampilkan informasi pada tiap-tiap lokasi bengkel yang ada pada peta.

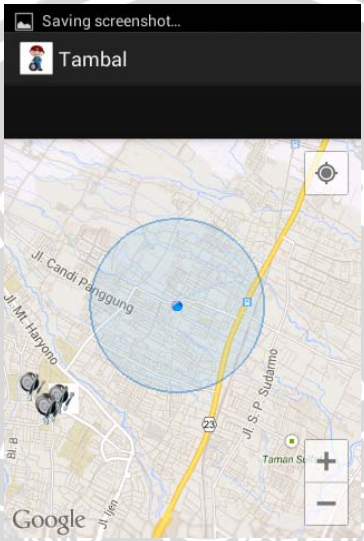
Dari implementasi basis data, implementasi *class* dan *source code* yang dijelaskan pada tabel 5.1, 5.2, 5.3, 5.4, 5.5, 5.6, 5.7 dan 5.8 didapatkan hasil dan tampilan aplikasi bengkel yang akan ditunjukkan dan jelaskan pada implementasi antarmuka.

5.4 Implementasi Antarmuka

Implementasi antarmuka aplikasi bengkel terdiri dari halaman utama yang menampilkan peta, lokasi pengguna saat itu dan menampilkan marker (lokasi bengkel) yang terdekat dengan pengguna. *Popup* (melihat informasi bengkel dan tombol *getdirection*). Pada tampilan *popup* menampilkan informasi bengkel tersebut beserta alamatnya, dan pada *popup* ini diberikan tombol *getdirection* (yang berfungsi bagi pengguna untuk mendapatkan rute dari lokasinya ke lokasi bengkel tujuan). Dan inplementasi antarmuka yang terakhir yaitu halaman *about*, pada halaman ini menampilkan informasi tentang aplikasi.

a. Halaman Utama

Halaman utama merupakan tampilan awal yang akan menampilkan halaman yang berisi peta, lokasi pengguna, dan marker lokasi bengkel ketika pengguna membuka aplikasi bengkel.



Gambar 5.1 Tampilan antarmuka halaman utama

Implementasi antarmuka aplikasi bengkel yang lainnya seperti *popup* (informasi bengkel) dan *getdirection* beserta *about* dapat dilihat pada lampiran 5.

5.5 Pengujian

Pengujian ini untuk mengetahui apakah sistem yang dibangun sudah benar sesuai dengan yang dibutuhkan. Pengujian validasi menggunakan metode pengujian *Black-Box*, karena tidak diperlukan konsentrasi terhadap alur jalannya algoritma program dan lebih ditekankan untuk menemukan kenyamanan antara kinerja sistem dengan daftar kebutuhan. Pada skripsi ini dilakukan pengujian validasi terhadap aplikasi bengkel.

5.5.1 Kasus Uji

5.5.1.1 Kasus Uji Pengguna

a. Kasus Uji Pembukaan Aplikasi Bengkel dan Kasus Uji Melihat Lokasi bengkel

Pada kasus uji ini menjelaskan tentang pengujian aplikasi pada pengguna, dan apakah aplikasi sudah sesuai dengan perancangan yang telah dibuat. Pada tabel 5.9 di bawah ini merupakan kasus uji pembukaan aplikasi bengkel.

Tabel 5.9 Kasus uji untuk pengujian validasi pembukaan aplikasi bengkel

Nama Kasus Uji	Kasus Uji Pembukaan Aplikasi Bengkel dan Kasus Uji Melihat Lokasi Bengkel
Objek Uji	Kebutuhan Fungsional (F01) dan (F0_2)
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa aplikasi dapat memenuhi kebutuhan fungsional untuk membuka aplikasi kemudian menampilkan halaman utama dan menampilkan lokasi bengkel.
Prosedur Uji	Pengguna menekan *icon* aplikasi bengkel.
Hasil yang Diharapkan	Aplikasi dapat menampilkan halaman utama, lokasi pengguna dan melihat marker bengkel terdekat serta dapat melihat marker bengkel yang lain.

Pada pengujian lokasi terdekat lokasi pengguna ke lokasi bengkel, dilakukan dua cara untuk mendapatkan hasil uji yang benar-benar valid. Pertama, dengan melakukan pengujian manual dengan cara melakukan penghitungan manual jarak dari pengguna ke lokasi bengkel terdekat dan lokasi bengkel yang lain. Dan yang kedua dilakukan pengujian langsung pada aplikasi bengkel dengan cara menjalankan aplikasi bengkel, apakah sudah sesuai hasil tampilannya dengan hasil penghitungan manual pada pengujian pertama.

Dikarenakan pada *google maps* belum tersedia tempat-tempat bengkel sebagaimana yang telah tersedia pada aplikasi, maka penguji melakukan pengujian menggunakan bantuan *google maps* dengan bertindak sebagai pengguna yang mengalami kerusakan kendaraan. Pertama penguji berada pada suatu titik lokasi untuk menentukan *latitude* dan *longitude* atau jarak keberadaanya dengan lokasi bengkel dengan bantuan *google maps*. Kedua, penguji datang ke tiap-tiap lokasi bengkel yang ada disekitar titik lokasi penguji,

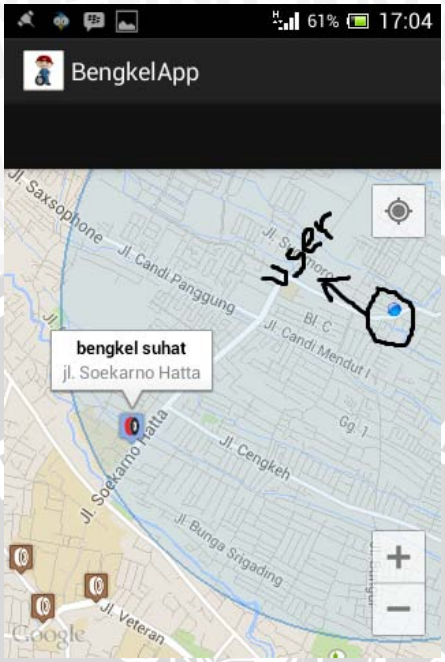
dan menentukan *latitude* serta *longitude* atau jarak dari lokasi keberadaanya ke lokasi bengkel tersebut.

Pada pengujian ini, penguji berada pada posisi / daerah blimbing. Bengkel yang pertama, yaitu bengkel A "(bengkel suhat)" yang berada di jl.Soekarno Hatta memliki jarak kurang lebih 2,1 kilo meter dari lokasi pengguna, sedangkan pada bengkel B "(bengkel sumber) yang berada di jl.Sumber sari memiliki jarak 3,9 kilo meter dari lokasi pengguna, dan pada bengkel uji yang terakhir yaitu bengkel C "(bengkel sigura) yang berada di jl.Sigura-gura memiliki jarak 4,1 kilo meter dari lokasi pengguna, seperti yang terlihat pada tabel 5.10 dibawah ini.

Tabel 5.10 Jarak lokasi pengguna ke lokasi bengkel

Bengkel		Jarak Lokasi pengguna ke Lokasi Bengkel
A	Bengkel Suhat	2,1 KM
B	Bengkel Sumber	3,9 KM
C	Bengkel Sigura	4,1 KM

Setelah jarak masing-masing bengkel dari lokasi pengguna telah didapatkan, dilakukan perbandingan antara bengkel A, B, dan C. Dapat disimpulkan bahwa bengkel A yaitu bengkel suhat yang berada di jl.Soekarno Hatta merupakan bengkel yang memiliki jarak paling dekat dengan lokasi pengguna. Setelah aplikasi bengkel dijalankan maka didapatkan hasil yang sama yaitu, bengkel suhat merupakan bengkel yang terdekat dengan lokasi pengguna saat itu. Seperti yang terlihat pada gambar 5.2 dibawah ini.



Gambar 5.2 bengkel yang terdekat dengan *user*

5.5.1.2 Hasil Uji

Dari hasil pengujian perhitungan manual dan pembukaan aplikasi secara langsung, maka dinyatakan bahwa aplikasi telah sesuai dan berjalan dengan apa yang diharapkan / valid seperti pada tabel 5.10 dibawah ini.

Tabel 5.10 Hasil pengujian

No.	Nama Kasus Uji	Hasil yang Diharapkan	Hasil yang Didapatkan	Status Validitas
1	Kasus Uji Pembukaan Aplikasi Bengkel	Aplikasi dapat menampilkan halaman utama (peta, lokasi pengguna, lokasi bengkel).	Aplikasi dapat menampilkan halaman utama (peta, lokasi pengguna, lokasi bengkel).	Valid
2	Kasus Uji Melihat Lokasi Bengkel	Aplikasi dapat menampilkan marker lokasi bengkel terdekat.	Hasil yang didapatkan : Aplikasi dapat menampilkan marker lokasi bengkel terdekat.	Valid

Hasil uji validasi yang ditunjukkan oleh Tabel 5.10 merupakan hasil uji yang dilakukan oleh 10 responden yang telah bersedia melakukan pengujian pada aplikasi bengkel.

b. Kasus Uji melihat informasi bengkel

Pada kasus uji ini menjelaskan tentang pengujian aplikasi pada pengguna, dan apakah aplikasi sudah sesuai dengan perancangan yang telah dibuat. Pada tabel 5.11 di bawah ini merupakan kasus uji untuk melihat informasi bengkel.

Tabel 5.11 Kasus uji untuk melihat informasi bengkel

Nama Kasus Uji	Kasus Uji Melihat Informasi Bengkel
Objek Uji	Kebutuhan Fungsional (F_03)
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa aplikasi dapat memenuhi kebutuhan fungsional untuk melihat informasi bengkel.
Prosedur Uji	Pengguna menekan marker bengkel.
Hasil yang Diharapkan	Aplikasi dapat menampilkan *popup* yang berisi informasi bengkel beserta tombol *getdirection*.

Pada pengujian melihat informasi bengkel ini, dilakukan untuk mendapatkan hasil uji yang benar-benar valid. Pertama, pengguna menekan marker bengkel, kemudian aplikasi akan menampilkan *popup* yang berisi informasi bengkel beserta tombol *getdirection* untuk mendapatkan rut eke lokasi bengkel.

c. Kasus Uji melihat rute bengkel

Pada kasus uji ini menjelaskan tentang pengujian aplikasi pada pengguna, dan apakah aplikasi sudah sesuai dengan perancangan yang telah dibuat. Pada tabel 5.12 di bawah ini merupakan kasus uji untuk melihat rute bengkel.

Tabel 5.12 Kasus uji untuk melihat rute bengkel

Nama Kasus Uji	Kasus Uji Melihat Rute Bengkel
Objek Uji	Kebutuhan Fungsional (F_04)
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa aplikasi dapat memenuhi kebutuhan fungsional untuk melihat rute lokasi bengkel dari keberadaan *user*.
Prosedur Uji	Pengguna menekan tombol *getdirection* yang ada pada *popup.*
Hasil yang Diharapkan	Aplikasi dapat menampilkan rute menuju bengkel terdekat yang dipilih dari keberadaan *user* saat itu.

Pada pengujian melihat rute bengkel ini, dilakukan untuk mendapatkan hasil uji yang benar-benar valid. Pertama pengguna menekan tombol *getdirection* yang ada pada *popup*, kemudian aplikasi akan menampilkan rute lokasi bengkel dari lokasi pengguna.

5.5.1.4 Hasil Uji

Dari hasil pengujian manual oleh pekerja pada pembukaan aplikasi secara langsung untuk melihat informasi dan rute bengkel, maka dinyatakan bahwa aplikasi telah sesuai dan berjalan dengan apa yang diharapkan / valid seperti pada tabel 5.13 dibawah ini.

Tabel 5.13 Hasil pengujian validasi

No.	Nama Kasus Uji	Hasil yang Diharapkan	Hasil yang Didapatkan	Status Validitas
1	Kasus Uji Melihat Informasi Bengkel	Aplikasi dapat menampilkan *popup* yang berisi informasi bengkel dan tombol *getdirection*	Hasil yang didapatkan : Aplikasi dapat menampilkan *popup* yang berisi informasi bengkel dan tombol *getdirection*	Valid
2	Kasus Uji Melihat RuteBengkel	Aplikasi dapat menampilkan rute bengkel dengan keberadaan pengguna	Hasil yang didapatkan : Aplikasi dapat menampilkan rute bengkel dengan keberadaan pengguna	Valid

Hasil uji validasi yang ditunjukkan oleh Tabel 5.13 merupakan hasil uji yang dilakukan oleh 10 responden yang telah bersedia melakukan pengujian pada aplikasi bengkel.

5.5.2.1 Kasus Uji Admin

a. Kasus Uji Tambah Data Lokasi Bengkel

Tabel 5.14 Kasus Uji Tambah Data Lokasi Bengkel

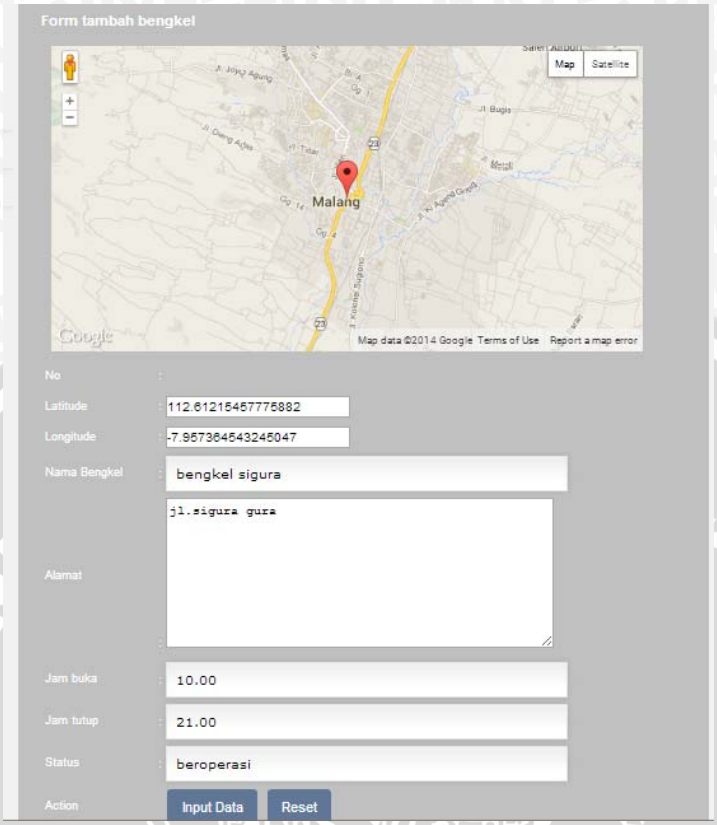
Nama Kasus Uji	Kasus Uji Tambah Data Lokasi Bengkel
Objek Uji	Kebutuhan Fungsional (F05) dan (F0_7)
Tujuan Pengujian	Pengujian dilakukan untuk memastikan bahwa *database* (server) / tempat penyimpanan data lokasi bengkel dapat memenuhi kebutuhan fungsional untuk memanipulasi data bengkel (tambah data) dan

	kemudian dapat ditampilkan pada aplikasi bengkel.
Prosedur Uji	Admin *login* pada *database* yang telah dirancang, dan melakukan manipulasi data di dalamnya (di dalam *database*).
Hasil yang Diharapkan	Aplikasi dapat menambahkan lokasi bengkel serta menampilkan hasil data lokasi bengkel terbaru yang telah dimanipulasi oleh admin pada *database* (server).

Pada kasus uji tambah data lokasi bengkel dilakukan beberapa tahap, yang pertama melakukan penambahan data pada *server database* admin dengan menekan tombol "*add new* bengkel". Seperti ditunjukkan pada gambar 5.2.

Gambar 5.3 Tambah data lokasi bengkel

Setelah itu akan keluar tampilan berikutnya untuk menentukan lokasi keberadaan bengkel baru yang akan ditambahkan. Pada gambar 5.3 dibawah ini merupakan jendela penambahan bengkel baru.



Gambar 5.4 *Add new* bengkel

Sebelum dilakukan penambahan bengkel baru, terlihat data bengkel seperti gambar 5.4 dibawah ini.

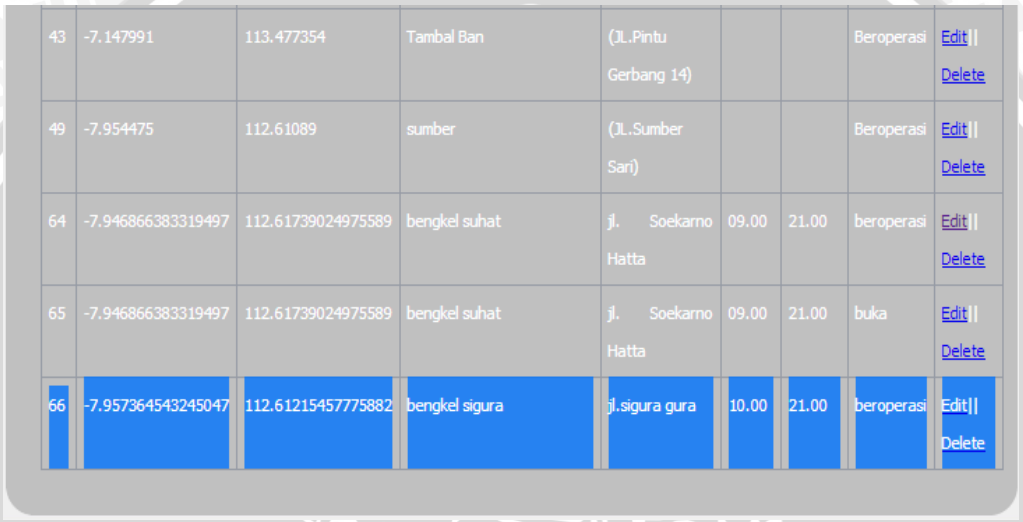
42	-7.15309	113.475455	Bengkel	(JL.Pintu Gerbang)			Beroperasi	Edit | Delete
43	-7.147991	113.477354	Tambal Ban	(JL.Pintu Gerbang 14)			Beroperasi	Edit | Delete
49	-7.954475	112.61089	sumber	(JL.Sumber Sari)			Beroperasi	Edit | Delete
64	-7.946866383319497	112.61739024975589	bengkel suhat	jl. Soekarno Hatta	09.00	21.00	beroperasi	Edit | Delete
65	-7.946866383319497	112.61739024975589	bengkel suhat	jl. Soekarno Hatta	09.00	21.00	buka	Edit | Delete

Gambar 5.5 Tampilan sebelum data bengkel baru ditambahkan

Dan setelah data diinputkan maka akan keluar tampilan seperti gambar 5.5 dan gambar 5.6 dibawah ini.

Gambar 5.6 Data bengkel telah berhasil ditambahkan

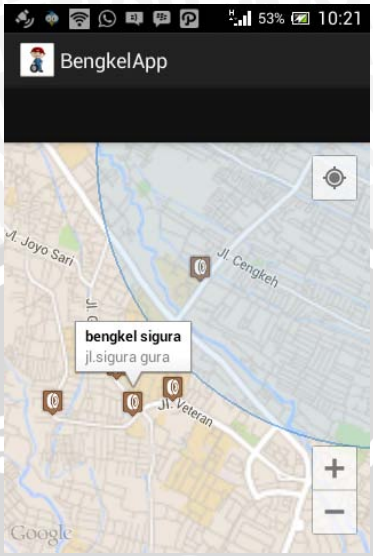
Dan hasil yang di dapat setelah data berhasil ditambahkan, maka akan ditampilkan gambar seperti gambar 5.6 dibawah ini.



43	-7.147991	113.477354	Tambal Ban	(JL.Pintu Gerbang 14)			Beroperasi	Edit|| Delete
49	-7.954475	112.61089	sumber	(JL.Sumber Sari)			Beroperasi	Edit|| Delete
64	-7.946866383319497	112.61739024975589	bengkel suhat	jl. Soekarno Hatta	09.00	21.00	beroperasi	Edit|| Delete
65	-7.946866383319497	112.61739024975589	bengkel suhat	jl. Soekarno Hatta	09.00	21.00	buka	Edit|| Delete
66	-7.957364543245047	112.61215457775882	bengkel sigura	jl.sigura gura	10.00	21.00	beroperasi	Edit|| Delete

Gambar 5.7 Data bengkel baru telah bertambah

Untuk mengetahui data bengkel baru pada aplikasi "bengkel" yang telah ditambahkan oleh admin pada *database / server* tersebut tadi, maka dapat kita jalankan aplikasi bengkel. Setelah dijalankan, didapat hasil seperti yang terlihat pada gambar 5.7 di bawah ini.



Gambar 5.8 Tampilan aplikasi bengkel setelah ditambah

5.5.2.2 Hasil Uji

Tabel 5.15 Hasil pengujian

No.	Nama Kasus Uji	Hasil yang Diharapkan	Hasil yang Didapatkan	Status Validitas
1	Kasus Uji Tambah Data Lokasi Bengkel	Aplikasi dapat menampilkan hasil data lokasi bengkel terbaru yang telah dimanipulasi oleh admin pada *database* (server).	Aplikasi dapat menampilkan hasil data lokasi bengkel terbaru.	Valid

5.3 Pengujian Non Fungsional

Pada pengujian non fungsional ini sistem akan diuji secara *usability*. pengujian *usability* sistem akan diuji dengan menggunakan kuisioner tentang kemudahan menggunakan sistem kepada para pengguna. Pengujian ini digunakan untuk melihat seberapa besar kepuasan pengguna dalam menggunakan aplikasi yang telah dikembangkan dan juga digunakan sebagai tolak ukur kesuksesan dalam mengembangkan suatu sistem. Kuisoner diberikan kepada 10 orang pengguna Android yang menginstal aplikasi bengkel. Hasil dari pengisian kuisioner dapat dilihat pada lampiran 1. Dari kuisioner yang telah diisi oleh para pengguna diperoleh hasil sebagai berikut :

Tabel 5.16 Tabel hasil kuisioner

Pertanyaan	Hasil
1. Bagaimanakah menurut Anda mangenai tampilan aplikasi yang dibuat?	80% pengguna menyatakan bahwa aplikasi memiliki tampilan yang sederhana dan mudah untuk dipahami. Karena ketika pengguna membuka aplikasi bengkel, maka sudah bisa menampilkan peta, lokasi pengguna dan lokasi bengkel terdekat.
2. Apakah aplikasi yang dikembangkan turut membantu pengguna ketika mengalami kerusakan di salah satu alat kendaraannya?	70% pengguna menyatakan bahwa aplikasi sangat membantu pengguna ketika mengalami kerusakan di salah satu kendaraanya karena dengan adanya aplikasi ini, mempercepat pelayanan.
3. Bagaimana pendapat anda jika aplikasi ini digunakan dan terus dikembangkan hingga menjadi aplikasi bengkel yang lebih kompleks?	50% pengguna menyatakan bahwa aplikasi dapat dikembangkan menjadi aplikasi bengkel yang lebih kompleks karena sangat membantu pengguna dalam pelayanan yang lebih cepat dan lebih baik. Dan 50% dari pengguna menyatakan bahwa aplikasi sudah cukup untuk aplikasi bengkel ini.
4. Apakah perlu adanya tutorial penggunaan untuk pengoperasional sistem?	95% pengguna menjawab tidak perlu adanya pelatihan untuk pengoperasionalan sistem.
5. Apakah aplikasi bengkel mudah dioperasikan?	95% pengguna menjawab aplikasi bengkel mudah dioperasikan.

5.4 Analisis Pengujian

Proses analisis bertujuan untuk mendapatkan kesimpulan dari hasil pengujian perangkat lunak bengkel yang telah dilakukan. Proses analisis mengacu pada dasar teori sesuai dengan hasil pengujian yang didapatkan. Analisis dilakukan terhadap hasil pengujian di setiap tahap pengujian. Pada tahap pengujian tampilan aplikasi yang terdiri dari 10 responden, 80% menilai bahwa tampilan aplikasi sederhana dan mudah dipahami. Sedangkan untuk pemenuhan kebutuhan oleh pengguna tentang aplikasi ini, 70% penggguna menilai bahwa aplikasi ini cukup membantu ketika mengalami kerusakan kendaraan, sedangkan 30% menilai bahwa tidak memerlukan aplikasi ini karena “hanya untuk mencari bengkel ataupun tambal ban saja masih harus membuka *handphone”*. Dan untuk tahap pengembangan aplikasi, 50% responden menilai bahwa aplikasi harus

dikembangkan sehingga lebih kompleks lagi, tetapi 50% yang lain menilai bahwa aplikasi ini tidak perlu dikembangkan lebih kompleks lagi karena masyarakat pamekasan masih belum sangat tergantung dengan adanya sistem atau perkembangan sistem informasi yang seperti ini.

Untuk tahap pengujian pengoprasian, 95% responden menilai bahwa aplikasi ini tidak memerlukan tutorial dikarenakan pengoprasiannya sudah sangat mudah yaitu, ketika pengguna menekan *icon* aplikasi bengkel maka pengguna sudah dapat melihat peta, lokasi keberadaanya, lokasi bengkel terdekat dan bengkel-bengkel yang lain, sedangkan 5% responden yang lain merasa perlu adanya tutorial untuk lebih bisa memahami jalannya aplikasi. Dan unutk tahap pengujian yang terakhir yaitu kemudahan pada aplikasi, 95% responden menilai bahwa aplikasi sangat mudah digunakan atau dioprasikan. Sedangkan 5% responden yang lain menilai bahwa aplikasi masih sedikit susah untuk dioprasikan karena kurang mengerti tentang teknologi.

5.4.1 Analisis Hasil Pengujian *Usability*

Proses analisis terhadap hasil pengujian *usability* dilakukan dengan melihat hasil kuisioner yang diberikan kepada 10 responden. Berdasarkan hasil pengujian *usability* pada tabel 5.13 dapat disimpulkan bahwa aplikasi bengkel dapat dioperasionalkan dengan mudah dan tidak memerlukan *tutorial* dalam penggunaannya.

BAB VI
PENUTUP

6.1 Kesimpulan

Berdasarkan hasil perancangan, implementasi dan pengujian yang dilakukan, maka diambil kesimpulan sebagai berikut :

1. Aplikasi bengkel ini merupakan aplikasi pencarian lokasi bengkel yang terdekat dengan pengguna.
2. Aplikasi bengkel dirancang sangat sederhana yaitu, ketika pengguna menekan *icon* aplikasi bengkel maka aplikasi sudah dapat menampilkan peta, lokasi pengguna dan lokasi bengkel terdekat serta lokasi bengkel yang lain dengan waktu kurang lebih 10 detik, sehingga memberikan kemudahan bagi pengguna disetiap pengoprasiannya.
3. Untuk menganalisis kebutuhan sistem, dilakukan survey langsung pada tiap-tiap lokasi bengkel. Dari hasil analisis tersebut dibuat desain perancangan untuk proses pembuatan aplikasi.
4. Berdasarkan hasil pengujian validasi sistem memenuhi kebutuhan fungsional yang di spesifikasikan.
5. Berdasarkan hasil kuisioner didapatkan bahwa aplikasi bengkel dapat dioperasionalkan dengan mudah dan tidak memerlukan *tutorial* dalam penggunaannya.

6.2 Saran

1. Untuk pengembang versi lebih lanjut sebaiknya tidak hanya diimplementasikan pada *platform* Android.
2. Menambahkan jumlah marker lokasi bengkel yang terdekat dengan keberadaan pengguna.

DAFTAR PUSTAKA

[AFD-13] Anton Firdaus, Dokumentasi *Google Maps* API, *https://developers.google.com/maps/documentation/android/*, Mei 2013.

[AHA-12] Akbarul Huda, Arif. 2012. 24 Jam Pintar Pemprograman Android. PT. Andi Yogyakarta. Juni 2013.

[CHA] Candra. 2013. Lab Studio Android + WebDeveloper Resource. 27 November.

[JKM-13] Joko, Munif. 2013. Dokumentasi Android. *http://developer.android.com*.Mei 2013.

[MML-09] Murphy M. L. 2009. *Beginning Android*. Apress. New York. Mei 2013.

[RRD-11] R Radifan. 2011. Perancangan dan Pembuatan Sistem Informasi Lokasi Friend Finder Berbasis GPS pada Sistem Operasi Android. Skripsi S-1. Institut Teknologi Sepuluh November. Surabaya.Mei 2013.

[SAP-13] Saputra, hendra. 2013. Aplikasi pencari lokasi. Akses dari *http://www.makemac.com/fieldtrip/*. Juni 2013.

[WNE-12] Winarno E. dan A. Zaki. 2012. Hacking & Programming dengan Android SDK untuk Advanced. PT Elex Media Komputindo. Jakarta. Mei 2013.

Lampiran 1

KUISIONER APLIKASI BENGKEL

Nama :

Jabatan :

Berilah jawaban pada pertanyaan (yang paling sesuai dengan kondisi Anda).

1. Bagaimanakah menurut Anda mangenai tampilan aplikasi yang dibuat?
   a. Sangat memuaskan
   b. Memuaskan
   c. Cukup
   d. Kurang memuaskan
   e. Tidak memuaskan

   Berikan alasan :

2. Apakah aplikasi yang dikembangkan turut membantu pengguna ketika mengalami kerusakan di salah satu alat kendaraannya?
   a. Sangat memuaskan
   b. Memuaskan
   c. Cukup
   d. Kurang memuaskan
   e. Tidak memuaskan

   Berikan alasan :

3. Bagaimana pendapat anda jika aplikasi ini digunakan dan terus dikembangkan hingga menjadi aplikasi bengkel yang lebih kompleks?
   a. Sangat Setuju
   b. TidakSetuju

   Berikan alasan :

4. Apakah perlu adanya tutorial penggunaan untuk pengoperasional sistem?
    a. Perlu
    b. Tidak perlu

    Berikan alasan :

5. Apakah aplikasi bengkel mudah dioperasikan?
    a. Sangat mudah
    b. Mudah
    c. Sedang
    d. Sulit
    e. Sangat sulit

    Berikan alasan :

Lampiran 2 Skenario *Use Case* melihat informasi bengkel

Nomor *Use Case*	F01
Nama *Use Case*	Melihat informasi bengkel
Prasyarat Konteks	Aplikasi telah terpasang pada *smartphone* Android pengguna
Tujuan Dalam Konteks	Mempermudah pengguna untuk melihat informasi bengkel
Prakondisi	Aplikasi sudah terpasang dan pengguna membuka aplikasi bengkel
Kondisi Akhir Sukses	Aplikasi menampilkan informasi bengkel
Kondisi Akhir Failed	-
Aktor	Pengguna
AlurUtama	Aktifitas
	3. Pengguna membuka aplikasi bengkel 4. Sistem menampilkan *popup* informasi bengkel dan tombol *get direction*

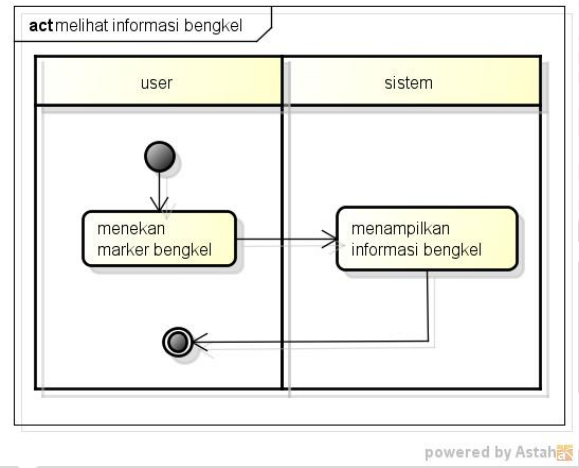
Lampiran 3 Skenario *Use Case* melihat rute bengkel

Nomor *Use Case*	F01
Nama *Use Case*	Melihat rute bengkel
Prasyarat Konteks	Aplikasi telah terpasang pada *smartphone* Android pengguna
Tujuan Dalam Konteks	Mempermudah pengguna untuk menuju bengkel yang ditentukan
Prakondisi	Aplikasi sudah terpasang dan pengguna membuka aplikasi bengkel
Kondisi Akhir Sukses	Aplikasi menampilkan rute bengkel
Kondisi Akhir Failed	-
Aktor	Pengguna
Alur Utama	Aktifitas
	1. Pengguna membuka aplikasi bengkel 2. Sistem menampilkan rute bengkel

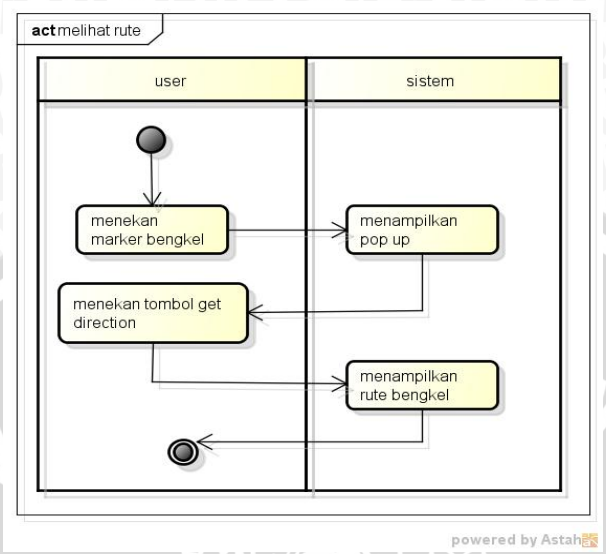
Lampiran 4 Skenario *Use Case* manipulasi data bengkel

Nomor *Use Case*	F02
Nama *Use Case*	Manipulasi data bengkel
Prasyarat Konteks	Admin berhasil login ke halaman home admin
Tujuan Dalam Konteks	Untuk melakukan tambah, *edit* dan hapus data bengkel
Prakondisi	Admin berhasil *login* dan melakukan manipulasi data bengkel
Kondisi Akhir Sukses	Sistem menampilkan hasil manipulasi data bengkel
Kondisi Akhir Failed	-
Aktor	Admin
Alur Utama	Aktifitas
	1. Pengguna *login* ke halaman admin 2. Sistem menampilkan hasil manipulasi data oleh admin

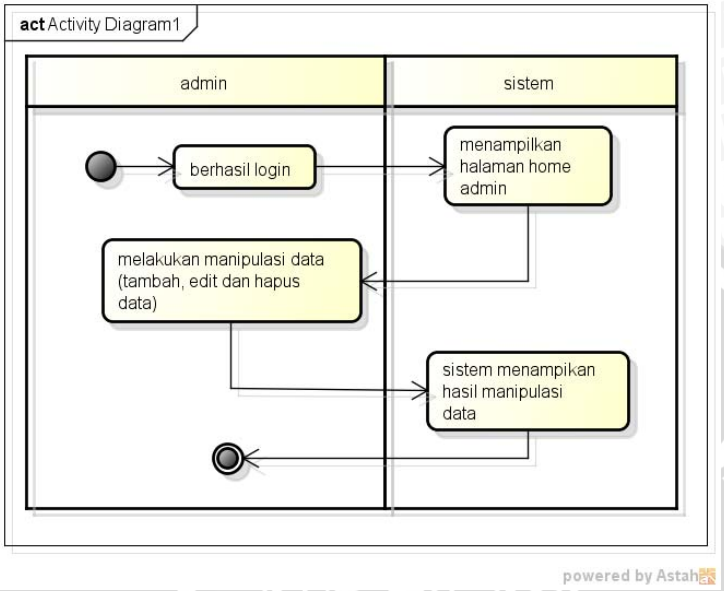
Lampiran 5 Aktivitas diagram melihat informasi bengkel



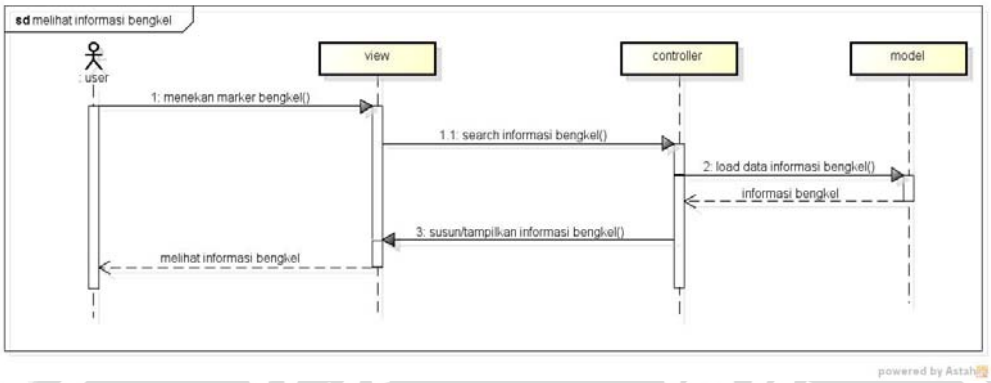
Lampiran 6 Aktivitas diagram melihat rute bengkel



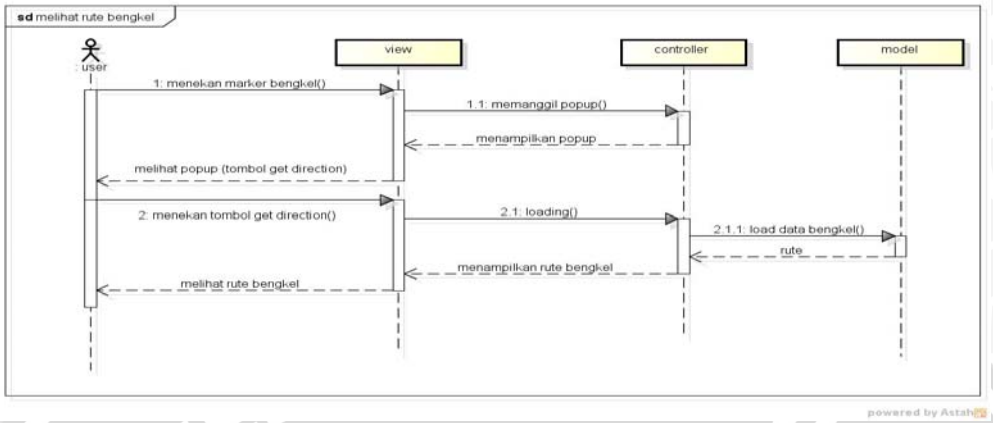
Lampiran 7 Aktivitas diagram manipulasi data bengkel



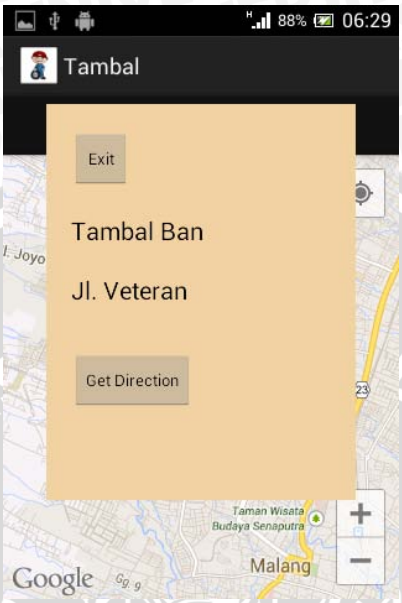
Lampiran 8 *Sequence diagram* melihat formasi bengkel



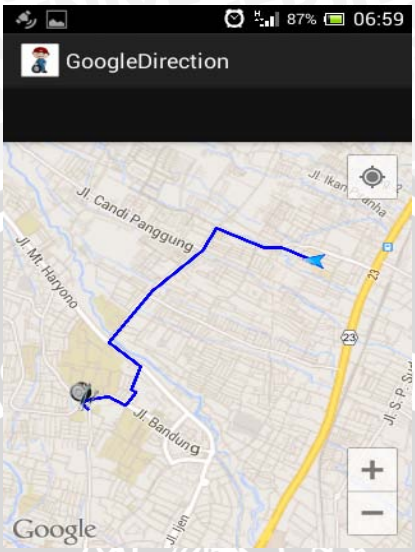
Lampiran 9 *Sequence diagram* melihat rute bengkel



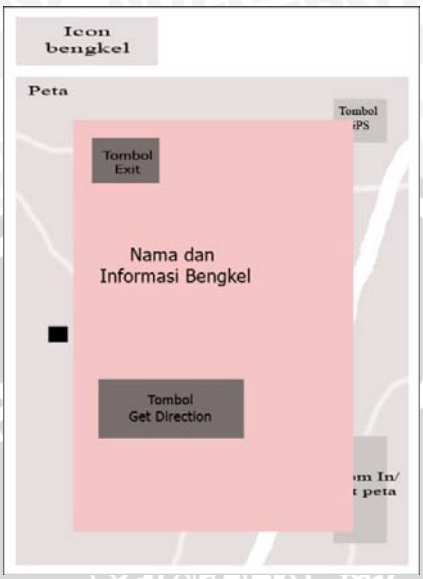
Lampiran 10 Tampilan antarmuka melihat informasi bengkel



Lampiran 11 Tampilan antarmuka melihat rute bengkel



Lampiran 12 Tampilan ketika melihat *popup* (informasi bengkel)



Lampiran 13 Tampilan ketika melihat rute bengkel